Knitro: An Integrated Package for Nonlinear Optimization

Richard H. Byrd[*] Jorge Nocedal[†] Richard A. Waltz[†]

February 6, 2006

Abstract

This paper describes Knitro 5.0, a C-package for nonlinear optimization that combines complementary approaches to nonlinear optimization to achieve robust performance over a wide range of application requirements. The package is designed for solving large-scale, smooth nonlinear programming problems, and it is also effective for the following special cases: unconstrained optimization, nonlinear systems of equations, least squares, and linear and quadratic programming. Various algorithmic options are available, including two interior methods and an active-set method. The package provides crossover techniques between algorithmic options as well as automatic selection of options and settings.

1 Introduction

Nonlinear programming problems are often difficult to solve. In spite of the rapid pace of algorithmic improvements, the most efficient algorithms available at present provide no guarantees of success or of fast performance over a range of applications. To complicate matters, the search for improved methods has led researchers to propose a variety of algorithms, each of which is typically implemented in a separate software package. To overcome the numerous difficulties that arise in practice, software developers have included a variety of options and heuristics to improve the chances of success. These packages are, however, constrained by the underlying algorithm, and as is well known, no single approach is uniformly successful in nonlinear optimization. The prospective user is thus faced with a difficult choice. Each code is unique in many ways: input and output formats, options and conventions. Thus there is a steep learning curve in trying to achieve the most effective use

[*]Department of Computer Science, University of Colorado, Boulder, CO 80309. This author was supported by Army Research Office Grants DAAG55-98-1-0176 and DAAD19-02-1-0407, and NSF grants CCR-0219190 and CHE-0205170.

[†]Department of Electrical and Computer Engineering, Northwestern University, Evanston, IL, 60208-3118, USA. These authors were supported by National Science Foundation grants CCR-0219438 and DMI-0422132, and Department of Energy grant DE-FG02-87ER25047-A004.

of a package. The availability of many codes through the NEOS Server `http://www-neos.mcs.anl.gov/` addresses only some of these issues.

The KNITRO software package aims to achieve greater flexibility and robustness through an integration of two powerful and complementary algorithmic approaches for nonlinear optimization: the interior-point approach and the active-set approach. The impressive success of an integrated approach of this sort for linear and integer programming, particularly over the past decade [21, 23], argues for a similar approach to be taken in nonlinear optimization. KNITRO is capable of applying features of an interior-point method or an active-set method — or possibly both — depending on problem characteristics. Within the interior-point approach, KNITRO provides two algorithms implementing distinct barrier approaches. One of the main challenges in the development of KNITRO has been the effective integration of the interior and active-set algorithms into a unified package, and the development of tools that exploit the power of our integrated approach.

The nonlinear programming formulation considered in this paper is:

$$\begin{aligned} \min_{x} \quad & f(x) && (1.1a) \\ \text{subject to} \quad & c_E(x) = 0 && (1.1b) \\ & c_I(x) \geq 0, && (1.1c) \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $c_E : \mathbb{R}^n \to \mathbb{R}^l$ and $c_I : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable functions. Problem (1.1) includes as special cases unconstrained optimization, systems of nonlinear equations, least squares problems, linear programs and quadratic programs. An important feature of the algorithms implemented in KNITRO is that they automatically reduce to effective algorithms for each of the simpler problem classes.

The quality and diversity of nonlinear optimization software has greatly improved during the last 10 years. Some of the established packages have matured, and new packages have emerged. SNOPT [18] and FILTERSQP [15] implement active-set sequential quadratic programming (SQP) methods. SNOPT uses a line search approach, and in its default setting, employs quasi-Newton approximations to the Hessian. FILTERSQP follows a trust region approach, with filter globalization, and makes use of second-derivative information. The MINOS [29] and LANCELOT [12] packages, which were the first widely available codes capable of solving problems with tens of thousands of variables and constraints, implement augmented Lagrangian methods. Another well established package is CONOPT [14], which offers reduced Hessian and SQP methods.

Most of the new packages are based on the interior-point approach. LOQO [33] implements a line search primal-dual algorithm that can be viewed as a direct extension of interior methods for linear and quadratic programming. The first release of KNITRO [6] offered a trust region interior-point algorithm employing a conjugate gradient iteration in the step computation; the second release added a line search interior algorithm that is safeguarded by the trust region approach [38]. BARNLP [2] and IPOPT [36] implement line search interior-point approaches; IPOPT uses a filter globalization and includes a feasibility restoration phase. MOSEK [1] is a primal-dual interior-point method for *convex* optimization, and PENNON [25] follows an augmented Lagrangian approach.

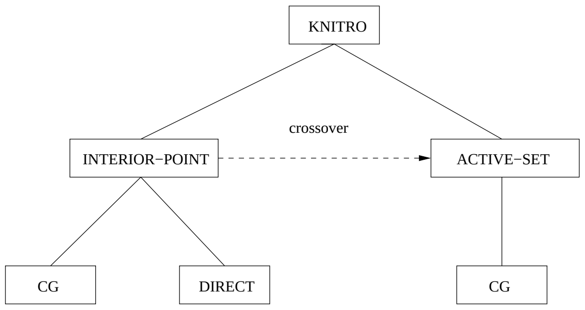

Figure 1: The main algorithmic options in the Knitro 5.0 package.

New active-set methods based on Sequential Linear-Quadratic Programming (SLQP) have recently been studied by Chin and Fletcher [9] and Byrd et al. [5]. Unlike SQP methods, which combine the active-set identification and the step computation in one quadratic subproblem, SLQP methods decouple these tasks into two subproblems. The active-set algorithm in Knitro, implements the SLQP method described in [5].

Interior-point and active-set methods offer competing state-of-the-art approaches for solving nonlinear optimization problems — each with its own set of advantages. Benchmarking studies [13, 28] have tried to identify the classes of problems for which each approach is best suited, but the rapid pace of software development makes it difficult to arrive at concrete conclusions at this time. We take the view that interior-point and active-set methods will both be needed in the years to come.

2 Overview of the Package

Knitro 5.0 is a C-package for solving nonlinear optimization problems. It is designed for large-scale applications, but it is also effective on small and medium scale problems. A great deal of attention has been given to the performance of the Knitro algorithms on simpler classes of problems such as systems of nonlinear equations and unconstrained problems because these tasks are crucial in the solution of nonlinear programming problems. We have also ensured that the algorithms are fast and reliable on linear and quadratic programming problems. A schematic view of the Knitro package is given in Figure 1.

In Figure 1 the nomenclature CG reflects the fact that the algorithmic step is computed using an iterative conjugate gradient approach, while DIRECT implies that the step is (usually) computed via a direct factorization of a linear system. As the figure suggests, the software design will enable the addition of future options in the package such as a DIRECT version of the active-set algorithm. Throughout the remainder of this paper we will refer to the implementations of the CG and direct interior-point algorithms in Knitro as Interior/CG and Interior/Direct, and the active-set algorithm implementation will be called Active.

In the following sections we give an outline of the algorithms implemented in Knitro. The descriptions are inevitably incomplete, since many additional features (such as second-order corrections, iterative refinement steps, resetting of parameters, and regularization procedures) are needed to achieve efficiency and robustness over a range of problems. Nevertheless, our outlines highlight the main features of the algorithms.

3 Interior-Point Methods

The interior (or barrier) methods implemented in KNITRO associate with (1.1) the barrier problem

$$\begin{aligned} \min_{x,s} \quad & f(x) - \mu \sum_{i=1}^{m} \log s_i && (3.1a) \\ \text{subject to} \quad & c_E(x) = 0 && (3.1b) \\ & c_I(x) - s = 0, && (3.1c) \end{aligned}$$

where s is a vector of slack variables and $\mu > 0$. The interior approach consists of finding (approximate) solutions of the barrier problem (3.1) for a sequence of positive barrier parameters $\{\mu_k\}$ that converges to zero.

The KKT conditions for (3.1) can be written as

$$\begin{aligned} \nabla f(x) - A_E{}^T(x)y - A_I{}^T(x)z &= 0 && (3.2a) \\ -\mu e + Sz &= 0 && (3.2b) \\ c_E(x) &= 0 && (3.2c) \\ c_I(x) - s &= 0, && (3.2d) \end{aligned}$$

where $e = (1, ..., 1)^T$, $S = \text{diag}(s_1, ..., s_m)$, A_E and A_I are the Jacobian matrices corresponding to the equality and inequality constraint vectors respectively, and y and z represent vectors of Lagrange multipliers. We also must have that $s, z \geq 0$. In the line search approach, we apply Newton's method to (3.2), backtracking if necessary so that the variables s, z remain positive, and so that the merit function is sufficiently reduced. In the trust region approach, we associate a quadratic program with (3.1) and let the step of the algorithm be an approximate solution of this quadratic subproblem. These two approaches are implemented, respectively, in the INTERIOR/DIRECT and INTERIOR/CG algorithms, and are described in more detail below.

The other major ingredient in interior methods is the procedure for choosing the sequence of barrier parameters $\{\mu_k\}$. Several options are provided in KNITRO. In the *Fiacco-McCormick/monotone approach*, the barrier parameter μ is held fixed for a series of iterations until the KKT conditions (3.2) are satisfied to some accuracy. An alternative is to use an *adaptive strategy* in which the barrier parameter is updated at every iteration. We have implemented the following adaptive update options: (i) the rule implemented in LOQO [33] based on the deviation of the minimum complementarity pair from the average; (ii) a probing strategy that uses Mehrotra's predictor step to select a target value for μ; (iii) a so-called quality-function approach; (iv) variants of option (ii) which possibly utilize safeguarded corrector steps. These rules are described and tested in Nocedal, Wäechter and Waltz [30]. Since it is not known at present which one is the most effective in practice, KNITRO allows the user to experiment with the barrier update strategies just mentioned.

To control the quality of the steps, both interior algorithms make use of the non-differentiable merit function

$$\phi_\nu(x,s) = f(x) - \mu \sum_{i=1}^{m} \log s_i + \nu \|c_E(x)\|_2 + \nu \|c_I(x) - s\|_2, \tag{3.3}$$

where $\nu > 0$. A step is acceptable only if it provides a sufficient decrease in ϕ_ν. Although it has been reported in the literature [22, 34] that merit functions of this type can interfere with rapid progress of the iteration, our experience indicates that the implementation described in Section 3.3 overcomes these difficulties. These observations are consistent with the results reported in Table 2 of Wächter and Biegler [36], which suggest that this merit function approach is as tolerant as a filter mechanism.

3.1 Algorithm I: KNITRO-INTERIOR/DIRECT

In this algorithm a typical iteration first computes a (primary) line search step using direct linear algebra. In order to obtain global convergence in the presence of non-convexity and Hessian or Jacobian singularities, the primary step may be replaced, under certain circumstances, by a safeguarding trust region step. INTERIOR-DIRECT is an implementation of the algorithm described in [38].

We begin by describing the (primary) line search step. Applying Newton's method to (3.2), in the variables x, s, y, z, gives

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & 0 & -A_E{}^T(x) & -A_I{}^T(x) \\ 0 & Z & 0 & S \\ A_E(x) & 0 & 0 & 0 \\ A_I(x) & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_s \\ d_y \\ d_z \end{bmatrix} = - \begin{bmatrix} \nabla f(x) - A_E{}^T(x)y - A_I{}^T(x)z \\ Sz - \mu e \\ c_E(x) \\ c_I(x) - s \end{bmatrix}, \tag{3.4}$$

where $\mathcal{L}$ denotes the Lagrangian

$$\mathcal{L}(x,s,y,z) = f(x) - y^T c_E(x) - z^T(c_I(x) - s). \tag{3.5}$$

If the inertia of the matrix in (3.4) is

$$(n+m, l+m, 0), \tag{3.6}$$

then the step d determined from (3.4) can be guaranteed to be a descent direction for the merit function (3.3). In this case, we compute the scalars

$$\alpha_s^{\max} = \max\{\alpha \in (0,1] : s + \alpha d_s \geq (1-\tau)s\}, \tag{3.7a}$$
$$\alpha_z^{\max} = \max\{\alpha \in (0,1] : z + \alpha d_z \geq (1-\tau)z\}, \tag{3.7b}$$

with $\tau = 0.995$. If $\min(\alpha_s^{\max}, \alpha_z^{\max})$ is not too small, we perform a backtracking line search that computes the steplengths

$$\alpha_s \in (0, \alpha_s^{\max}], \qquad \alpha_z \in (0, \alpha_z^{\max}], \tag{3.8}$$

providing sufficient decrease of the merit function (3.3). The new iterate is then defined as

$$x^+ = x + \alpha_s d_x, \qquad s^+ = s + \alpha_s d_s, \tag{3.9a}$$

$$y^+ = y + \alpha_z d_y, \qquad z^+ = z + \alpha_z d_z. \tag{3.9b}$$

On the other hand, if the inertia is not given by (3.6) or if the steplength α_s or α_z is less than a given threshold α_{min}, then the primary step d is rejected. In this case the algorithm reverts to the trust region method implemented in the INTERIOR/CG algorithm (see the next section) which is guaranteed to provide a successful step even in the presence of negative curvature or singularity.

The use of this safeguarding trust region step makes the KNITRO-INTERIOR/DIRECT algorithm distinct from other line search interior-point algorithms, such as BARNLP, IPOPT and LOQO, which modify the Hessian $\nabla^2_{xx}\mathcal{L}$ whenever the inertia condition (3.6) is not satisfied. We prefer to revert to a trust region iteration because this permits us to compute a step using a null-space approach, without modifying the Hessian $\nabla^2_{xx}\mathcal{L}$. An additional benefit of invoking the trust region step is that it guarantees progress in cases when the line search approach can fail [7, 35]. Since it is known that, when line search iterations converge to non-stationary points, the steplengths α_s or α_z in (3.9) converge to zero, we monitor these steplengths. If one of them is smaller than a given threshold, we discard the line search iteration (3.4)-(3.9) and replace it with the trust region step.

We outline the method in Algorithm 3.1. Here $D\phi_\nu(x, s; d)$ denotes the directional derivative of the merit function ϕ_ν along a direction d. The algorithm maintains a trust region radius Δ_k at every iteration, in case it needs to revert to the trust region approach.

The initial multipliers y_0, z_0 are computed as the least-squares solution of the system (3.2a)-(3.2b). When the line search step is discarded (the last If-Endif block in Algorithm 3.1) we compute one or more INTERIOR/CG steps (described in the following section) until one of them provides sufficient reduction in the merit function.

We assume in Algorithm 3.1 that we are using the Fiacco-McCormick/monotone approach for updating the barrier parameter μ. However, this algorithm is easily modified to implement the adaptive barrier update strategies discussed at the beginning of Section 3. In this case, there is no barrier stop test and the barrier parameter μ is updated at every iteration using some adaptive rule (which could cause μ to increase or decrease).

3.2 Algorithmic Option II: KNITRO-INTERIOR/CG

The second interior algorithm implemented in KNITRO computes steps by using a quadratic model and trust regions. This formulation allows great freedom in the choice of the Hessian and provides a mechanism for coping with Jacobian and Hessian singularities. The price for this flexibility is a more complex iteration than in the line search approach. INTERIOR/CG is an implementation of the algorithm described in [6], which is based on the approach described and analyzed in [3].

Algorithm 3.1: KNITRO-INTERIOR/DIRECT

```
Choose $x_0$, $s_0 > 0$, and the parameters $0 < \eta$, and $0 < \alpha_{min} < 1$. Compute initial
values for the multipliers $y_0$, $z_0 > 0$, the trust-region radius $\Delta_0 > 0$, and the barrier
parameter $\mu > 0$. Set $k = 0$.
**Repeat** until a stopping test for the nonlinear program (1.1) is satisfied:
    **Repeat** until the perturbed KKT conditions (3.2) are approximately satisfied:
        Factor the primal-dual system (3.4) and record the number neig
        of negative eigenvalues of its coefficient matrix.
        Set LineSearch = False.
        **If** neig $\leq l + m$
            Solve (3.4) to obtain the search direction $d = (d_x, d_s, d_y, d_z)$.
            Define $w = (x_k, s_k)$ and $d_w = (d_x, d_s)$.
            Compute $\alpha_s^{\max}, \alpha_z^{\max}$ by (3.7).
            **If** $\min\{\alpha_s^{\max}, \alpha_z^{\max}\} > \alpha_{min}$,
                Update the penalty parameter $\nu_k$ (see Section 3.3).
                Compute a steplength $\alpha_s = \bar{\alpha}\alpha_s^{\max}, \bar{\alpha} \in (0,1]$ such that
                $\phi_\nu(w + \alpha_s d_w) \leq \phi_\nu(w) + \eta\alpha_s D\phi_\nu(w; d_w)$.
                **If** $\alpha_s > \alpha_{min}$,
                    Set $\alpha_z = \bar{\alpha}\alpha_z^{\max}$.
                    Set $(x_{k+1}, s_{k+1}, y_{k+1}, z_{k+1})$ by (3.9).
                    Set LineSearch = True.
                **Endif**
            **Endif**
        **Endif**
        **If** LineSearch == False,
            Compute $(x_{k+1}, s_{k+1}, y_{k+1}, z_{k+1})$ using the INTERIOR/CG algorithm
            of Section 3.2.
        **Endif**
        Compute $\Delta_{k+1}$.
        Set $k \leftarrow k + 1$.
    **End**
    Choose a smaller value for the barrier parameter $\mu$.
**End**
```

To motivate the INTERIOR/CG algorithm, we first note that the barrier problem (3.1) is an equality-constrained optimization problem and can be solved by using a sequential quadratic programming method with trust regions. A straightforward application of SQP techniques to the barrier problem leads, however, to inefficient steps that tend to violate the positivity of the slack variables and are frequently cut short by the trust-region constraint. To overcome this problem, we design the following SQP method specifically tailored to the barrier problem.

At the current iterate (x_k, s_k), and for a given barrier parameter μ, we first compute Lagrange multiplier estimates (y_k, z_k) and then compute a step $d = (d_x, d_s)$ that aims to solve the subproblem,

$$\min_{d_x, d_s} \quad \nabla f(x_k)^T d_x + \frac{1}{2} d_x^T \nabla_{xx}^2 \mathcal{L}(x_k, s_k, y_k, z_k) d_x - \mu e^T S_k^{-1} d_s + \frac{1}{2} d_s^T \Sigma_k d_s \tag{3.10a}$$

$$\text{subject to} \quad A_E(x_k) d_x + c_E(x_k) = r_E \tag{3.10b}$$

$$A_I(x_k) d_x - d_s + c_I(x_k) - s_k = r_I \tag{3.10c}$$

$$\|d_x, S_k^{-1} d_s\|_2 \le \Delta_k \tag{3.10d}$$

$$d_s \ge -\tau s, \tag{3.10e}$$

where $\Sigma_k = S_k^{-1} Z_k$ and $\tau = 0.995$. Ideally, we would like to set $r = (r_E, r_I) = 0$, but since this could cause the constraints to be incompatible or produce poor steps, we choose r as the smallest vector such that (3.10b)-(3.10d) are consistent (with some margin). This computation is described in more detail below.

We can motivate the choice of the objective (3.10a) by noting that the first-order optimality conditions of (3.10a)-(3.10c) are given by (3.2) (with the second block of equations scaled by S^{-1}). The steps computed by using (3.10) are thus related to those of the line search algorithm described in the previous section. The trust-region constraint (3.10d) guarantees that (3.10) has a finite solution even when $\nabla_{xx}^2 \mathcal{L}(x_k, s_k, y_k, z_k)$ is not positive definite. Therefore the Hessian need never be modified. The scaling S_k^{-1} in the trust-region constraint is crucial; its effect on the iteration will be discussed later on.

Step Computation

The subproblem (3.10) is difficult to minimize exactly because of the presence of the nonlinear constraint (3.10d) and the bounds (3.10e). We can, however, compute useful inexact solutions, at a moderate cost. To do so, KNITRO follows a null-space approach in which the step d is the sum of a *normal step* v that attempts to satisfy the linear constraints (3.10b)-(3.10c) (with $r = 0$) as well as possible subject to a trust region, and a *tangential step* that lies on the tangent space of the constraints and that tries to achieve optimality.

To compute the normal step $v = (v_x, v_s)$, we formulate the following subproblem:

$$\min_{v} \quad \|A_E v_x + c_E\|_2^2 + \|A_I v_x - v_s + c_I - s\|_2^2 \tag{3.11a}$$

$$\text{subject to} \quad \|(v_x, S^{-1} v_s)\|_2 \le 0.8\Delta. \tag{3.11b}$$

(Here and below we omit the arguments of the functions for simplicity.) We compute an inexact solution of this problem using a dogleg approach, which minimizes (3.11a) along a piecewise linear path composed of a steepest descent step in the norm used in (3.11b) and a minimum-norm Newton step with respect to the same norm. The scaling $S^{-1}v_s$ in the norm tends to limit the extent to which the bounds on the slack variables are violated.

Once the normal step v is computed, we define the vectors r_E and r_I in (3.10b)-(3.10c) as the residuals in the normal step computation, namely,

$$r_E = A_E v_x + c_E, \qquad r_I = A_I v_x - v_s + (c_I - s).$$

Having computed the normal step (v_x, v_s), the subproblem (3.10) can therefore be written as

$$\begin{aligned} \min_{d_x, d_s} \quad & \nabla f^T d_x - \mu e^T S^{-1} d_s + \frac{1}{2}(d_x^T \nabla^2_{xx}\mathcal{L} d_x + d_s^T \Sigma d_s) && \text{(3.12a)} \\ \text{subject to} \quad & A_E d_x = A_E v_x && \text{(3.12b)} \\ & A_I d_x - d_s = A_I v_x - v_s && \text{(3.12c)} \\ & \|(d_x, S^{-1} d_s)\|_2 \le \Delta, && \text{(3.12d)} \end{aligned}$$

which we refer to as the tangential subproblem. To find an approximate solution d of (3.12), we first introduce the scaling

$$\tilde{d}_s \leftarrow S^{-1} d_s, \tag{3.13}$$

which transforms (3.12d) into a sphere. Then we apply the projected conjugate gradient (CG) method of Section 5 to the transformed quadratic program, iterating in the linear manifold defined by (3.12b)-(3.12c). During the solution by CG, we use a Steihaug strategy, monitoring the satisfaction of the trust-region constraint (3.12d), and stopping if the boundary of this region is reached or if negative curvature is detected. Finally, we truncate the step d if necessary in order to satisfy (3.10e).

We outline this interior method in Algorithm 3.2. Here

$$\text{ared}(d) = \phi_\nu(x, s) - \phi_\nu(x + d_x, s + d_s) \tag{3.14}$$

is the actual reduction in the merit function, and the predicted reduction, pred(d), is defined by (3.15),(3.16).

The multiplier estimates (y_k, z_k) are computed by a least squares approximation to the equations (3.2a)-(3.2b) at x_k, and shifted to ensure positivity of z_k. The barrier stop tolerance can be defined as $\epsilon_\mu = \mu$. As with the INTERIOR/DIRECT algorithm, this algorithm is easily modified to implement adaptive barrier update strategies.

The interior-point method outlined in Algorithm 3.2 is asymptotically equivalent to standard line search interior methods, but it is significantly different in two respects. First, it is not a fully primal-dual method in the sense that multipliers are computed as a function of the primal variables (x, s) — as opposed to the formulation (3.4) in which primal and dual variables are computed simultaneously from their previous values. Second, the trust-region method uses a scaling of the variables that discourages moves toward the boundary of the feasible region. This causes the algorithm to generate steps that can be very different from those produced by a line search method.

```
Algorithm 3.2: KNITRO-INTERIOR/CG
Choose parameter $\eta > 0$. Choose initial values for $\mu > 0$, $x_0$, $s_0 > 0$ and $\Delta_0 > 0$.
Set $k = 0$.
Repeat until a stopping test for the nonlinear program (1.1) is satisfied:
    Repeat until the perturbed KKT conditions (3.2) are approximately satisfied:
        Compute the normal step $v_k = (v_x, v_s)$.
        Compute Lagrange multipliers $y_k, z_k > 0$.
        Compute the total step $d_k$ by applying the projected CG method to
            (3.12$a$)-(3.12$c$) (see Section 5).
        Update the penalty parameter $\nu_k$ (see Section 3.3).
        Compute $\text{ared}_k(d_k)$ by (3.14) and $\text{pred}_k(d_k)$ by (3.16).
        If $\text{ared}_k(d_k) \geq \eta \text{pred}_k(d_k)$
            Set $x_{k+1} = x_k + d_x$, $s_{k+1} = s_k + d_s$, and update $\Delta_{k+1}$.
        Else
            Set $x_{k+1} = x_k$, $s_{k+1} = s_k$, and choose $\Delta_{k+1} < \Delta_k$.
        Endif
        Set $k \leftarrow k + 1$.
    End
    Choose a smaller value for the barrier parameter $\mu$ .
End
```

3.3 Merit Function

The role of the merit function (3.3) is to determine whether a step is productive and should be accepted. Our numerical experience has shown that the choice of the merit parameter ν plays a crucial role in the efficiency of the algorithm. Both interior-point methods in KNITRO choose ν at every iteration so that the decrease in a quadratic model of the merit function produced by a step d is proportional to the product of ν times the decrease in linearized constraints.

To be more specific, suppose that either the INTERIOR/DIRECT or INTERIOR/CG algorithm has produced a step d. We define the following linear/quadratic model of the merit function ϕ_ν:

$$Q_\nu(d) = \nabla f^T d_x - \mu e^T S^{-1} d_s + \frac{\sigma}{2}\left(d_x^T \nabla_{xx}^2 \mathcal{L} d_x + d_s^T \Sigma d_s\right) + \nu\left(m(0) - m(d)\right), \tag{3.15}$$

where

$$m(d) = \left\| \begin{bmatrix} A_E d_x + c_E \\ A_I d_x - d_s + c_I - s \end{bmatrix} \right\|_2,$$

denotes the first-order violation of the constraints, and σ is a parameter to be discussed below. We also define the predicted decrease in the merit function as

$$\text{pred}(d) = Q_\nu(0) - Q_\nu(d). \tag{3.16}$$

In all cases we choose the penalty parameter ν large enough such that

$$\text{pred}(d) \geq \rho\nu[m(0) - m(d)], \tag{3.17}$$

for some parameter $0 < \rho < 1$ (e.g. $\rho = 0.1$). If the value of ν from the previous iteration satisfies (3.17), it is left unchanged, otherwise ν is increased so that it satisfies this inequality with some margin. Condition (3.17) is standard for trust region methods, but not for line search methods, where it may require ν to be larger than is needed to simply provide a descent direction. As shown in [38] this stronger condition can improve performance of the line search iteration.

For a trust region method, such as that implemented in INTERIOR/CG, we set $\sigma = 1$ in (3.15) because these methods can deal well with indefiniteness of the Hessian. A line search method, on the other hand, does not always produce a descent direction for the merit function if the model on which it is based is not convex. Therefore in the INTERIOR/DIRECT algorithm we define σ as

$$\sigma = \begin{cases} 1 & \text{if } d_x^T \nabla_{xx}^2 \mathcal{L} d_x + d_s^T \Sigma d_s > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.18}$$

This choice of σ guarantees the directional derivative of ϕ_ν in the direction d is negative.

4 Active-set Sequential Linear-Quadratic Programming

The active-set method implemented in Knitro does not follow an SQP approach because, in our view, the cost of solving generally constrained quadratic programming subproblems imposes a limitation on the size of problems that can be solved in practice. In addition, the incorporation of second derivative information in SQP methods has proved to be difficult.

We use, instead a sequential linear-quadratic programming (SLQP) method [5, 9, 16] that computes a step in two stages, each of which scales up well with the number of variables. First, a linear program (LP) is solved to identify a working set. This is followed by an equality constrained quadratic programming (EQP) phase in which the constraints in the working set $\mathcal{W}$ are imposed as equalities. The total step of the algorithm is a combination of the steps obtained in the linear programming and equality constrained phases.

To achieve progress on both feasibility and optimality, the algorithm is designed to reduce the ℓ_1 penalty function,

$$P(x;\nu) = f(x) + \nu \sum_{i\in\mathcal{E}} |c_i(x)| + \nu \sum_{i\in\mathcal{I}} (\max(0, -c_i(x)), \tag{4.1}$$

where $c_i,\ i \in \mathcal{E}$, denote the components of the vector c_E, and similarly for c_I. The penalty parameter ν is chosen by an adaptive procedure described below.

An appealing feature of the SLQP algorithm is that established techniques for solving large-scale versions of the LP and EQP subproblems are readily available. Modern LP software is capable of solving problems with more than a million variables and constraints, and the solution of an EQP can be performed efficiently using the projected conjugate gradient iteration discussed in Section 5. We now outline the SLQP approach implemented in Knitro-Active. This algorithm is an implementation of the algorithm Slique described in [5].

4.1 Algorithm III: Knitro-Active

In the LP phase, given an estimate x_k of the solution of the nonlinear program (1.1), we would like to solve

$$\begin{aligned} \min_d \quad & \nabla f(x_k)^T d && (4.2a)\\ \text{subject to} \quad & c_i(x_k) + \nabla c_i(x_k)^T d = 0, \quad i \in \mathcal{E} && (4.2b)\\ & c_i(x_k) + \nabla c_i(x_k)^T d \geq 0, \quad i \in \mathcal{I} && (4.2c)\\ & \|d\|_\infty \leq \Delta_k^{\text{LP}}, && (4.2d) \end{aligned}$$

with $\Delta_k^{\text{LP}} > 0$. (Note that (4.2) differs from the subproblem used in SQP methods only in that the latter include a term of the form $\frac{1}{2}d^T H d$ in (4.2a), where H is an approximation to the Hessian of the Lagrangian of the nonlinear program.) Since the constraints of (4.2) may be inconsistent, we solve instead the ℓ_1 penalty reformulation of (4.2) given by

$$\min_d \quad l_\nu(d) \overset{\text{def}}{=} \nabla f(x_k)^T d + \nu_k \sum_{i\in\mathcal{E}} |c_i(x_k) + \nabla c_i(x_k)^T d|$$

$$
\begin{aligned}
& & +\nu_k \sum_{i\in\mathcal{I}} \max\left(0, -c_i(x_k) - \nabla c_i(x_k)^T d\right) \qquad & (4.3a)\\
&\text{subject to} & \|d\|_\infty \le \Delta_k^{\text{LP}}. \qquad & (4.3b)
\end{aligned}
$$

The solution of this linear program, which we denote by d^{LP}, is computed by the simplex method so as to obtain an accurate estimate of the optimal active set.

Based on this solution, we define the working set $\mathcal{W}$ as some linearly independent subset of the active set $\mathcal{A}$ at the LP solution, which is defined as

$$\mathcal{A}(d^{\text{LP}}) = \{i \in \mathcal{E} \mid c_i(x_k) + \nabla c_i(x_k)^T d^{\text{LP}} = 0\} \cup \{i \in \mathcal{I} \mid c_i(x_k) + \nabla c_i(x_k)^T d^{\text{LP}} = 0\}.$$

Likewise, we define the set $\mathcal{V}$ of violated constraints as

$$\mathcal{V}(d^{\text{LP}}) = \{i \in \mathcal{E} \mid c_i(x_k) + \nabla c_i(x_k)^T d^{\text{LP}} \neq 0\} \cup \{i \in \mathcal{I} \mid c_i(x_k) + \nabla c_i(x_k)^T d^{\text{LP}} < 0\}.$$

To ensure that the algorithm makes progress on the penalty function P, we define the *Cauchy step*,

$$d^{\text{C}} = \alpha^{\text{LP}} d^{\text{LP}}, \qquad (4.4)$$

where $\alpha^{\text{LP}} \in (0,1]$ is a steplength that provides sufficient decrease in the following (piece-wise) quadratic model of the penalty function $P(x;\nu)$:

$$q_k(d) = l_\nu(d) + \frac{1}{2} d^T H(x_k, \lambda_k) d. \qquad (4.5)$$

Here H is the Hessian of the Lagrangian or an approximation to it, and $l_\nu(d)$ is defined in (4.3a).

Given the working set $\mathcal{W}_k$, we now solve an equality constrained quadratic program (EQP) treating the constraints in $\mathcal{W}_k$ as equalities and ignoring all other constraints. This gives the subproblem

$$
\begin{aligned}
&\min_d & \frac{1}{2} d^T H(x_k,\lambda_k) d + \left(\nabla f(x_k) + \nu_k \sum_{i\in\mathcal{V}} \gamma_i \nabla c_i(x_k)\right)^T d \qquad & (4.6a)\\
&\text{subject to} & c_i(x_k) + \nabla c_i(x_k)^T d = 0, \quad i \in \mathcal{E} \cap \mathcal{W}_k \qquad & (4.6b)\\
& & c_i(x_k) + \nabla c_i(x_k)^T d = 0, \quad i \in \mathcal{I} \cap \mathcal{W}_k \qquad & (4.6c)\\
& & \|d\|_2 \le \Delta_k, \qquad & (4.6d)
\end{aligned}
$$

where γ_i is the algebraic sign of the violated i-th constraint. Note that the trust region (4.6d) is spherical, and is distinct from the trust region Δ^{LP} used in (4.2d). Problem (4.6) is solved for the vector d^{Q} by applying the projected conjugated gradient procedure described in Section 5. The total step d of the SLQP method is given by

$$d = d^{\text{C}} + \alpha^{\text{Q}}(d^{\text{Q}} - d^{\text{C}}),$$

where $\alpha^{\text{Q}} \in [0,1]$ is a steplength that approximately minimizes the model function (4.5).

```
Algorithm 4.1: Knitro-Active
Initial data: $x_0$, $\Delta_0 > 0$, $\Delta_0^{\text{LP}} > 0$, $0 < \eta < 1$. Set $k = 0$.
Repeat until a stopping test for the nonlinear program (1.1) is satisfied:
    LP point. Update the penalty parameter $\nu_k$ and solve the LP (4.3)
        to obtain the step $d_k^{\text{LP}}$, and working set $\mathcal{W}_k$.
    Cauchy point. Compute $\alpha_k^{\text{LP}} \in (0,1]$ as an approximate minimizer of
        $q(\alpha d_k^{\text{LP}})$ such that $\alpha_k^{\text{LP}} \|d_k^{\text{LP}}\| \leq \Delta_k$. Set $d_k^{\text{C}} = \alpha_k^{\text{LP}} d_k^{\text{LP}}$ .
    EQP point. Compute $d_k^{\text{Q}}$ by solving the EQP (4.6).
        Define $d_k^{\text{CE}} = d_k^{\text{Q}} - d_k^{\text{C}}$ as the segment leading from the Cauchy point
        to the EQP point.
    Trial point. Compute $\alpha_k^{\text{Q}} \in [0,1]$ as an approximate minimizer of
        $q(d_k^{\text{C}} + \alpha d_k^{\text{CE}})$. Set $d_k = d_k^{\text{C}} + \alpha_k^{\text{Q}} d_k^{\text{CE}}$ and $x_{\text{T}} = x_k + d_k$.
    Step Acceptance. Compute
            $\rho_k = (P(x_k;\nu_k) - P(x_{\text{T}};\nu_k))/(q_k(0) - q_k(d_k))$.
        If $\rho_k \geq \eta$, set $x_{k+1} \leftarrow x_{\text{T}}$, otherwise set $x_{k+1} \leftarrow x_k$.
    Update $\Delta_{k+1}^{\text{LP}}$ and $\Delta_{k+1}$. Set $k \leftarrow k + 1$.
End
```

The trust region radius Δ_k for the EQP phase is updated based on the ratio ρ_k, following standard trust region update strategies. The choice of Δ_{k+1}^{LP} is based on an effort to generate a good working set. In our implementation, Δ_{k+1}^{LP} is set to be a little larger than the total step d_k, subject to some other restrictions, as described in [5]. The multiplier estimates λ_k used in the Hessian are least squares estimates using the working set $\mathcal{W}_k$, and modified so that $\lambda_i \geq 0$ for $i \in \mathcal{I}$.

Penalty Parameter Update Strategy.

A novel feature of our SLQP algorithm is the procedure for updating the penalty parameter. Unlike most strategies proposed in the literature [11], which hold the penalty parameter ν fixed for a series of iterations and only update it if insufficient progress toward feasibility is made, our algorithm chooses an appropriate value of ν at each iteration. This selection takes place during the linear programming phase, as we now explain.

We define a (piecewise) linear model of constraint violation at a point x_k by

$$m_k(d) = \sum_{i \in \mathcal{E}} |c_i(x_k) + \nabla c_i(x_k)^T d| + \sum_{i \in \mathcal{I}} \max\left(0, -c_i(x_k) - \nabla c_i(x_k)^T d\right), \tag{4.7}$$

so that the objective (4.3) of the LP subproblem can be written as

$$l_\nu(d) = \nabla f(x_k)^T d + \nu_k \, m_k(d). \tag{4.8}$$

Given a value ν_k, we write the solution of the LP problem (4.3) as $d^{\text{LP}}(\nu_k)$ to stress its dependence on the penalty parameter. Likewise, $d^{\text{LP}}(\nu_\infty)$ denotes the minimizer of $m_k(d)$ subject to the trust region constraint (4.3*b*). The following algorithm describes the computation of the LP step d_k^{LP} and the penalty parameter ν_k.

Algorithm Penalty Update. LP Step and Penalty Update Strategy.

```
Initial data: $x_k, \nu_{k-1} > 0$, $\Delta_k^{\text{LP}} > 0$, and parameters $\epsilon_1, \epsilon_2 \in (0,1)$.
Solve the subproblem (4.3) with $\nu = \nu_{k-1}$ to obtain $d^{\text{LP}}(\nu_{k-1})$.
If $m_k(d^{\text{LP}}(\nu_{k-1})) = 0$
    Set $\nu^+ \leftarrow \nu_{k-1}$.
Else compute $d^{\text{LP}}(\nu_\infty)$
    If $m_k(d^{\text{LP}}(\nu_\infty)) = 0$
        Find $\nu^+ > \nu_{k-1}$ such that $m_k(d^{\text{LP}}(\nu^+)) = 0$.
    Else
        Find $\nu^+ \geq \nu_{k-1}$ such that
            $m_k(0) - m_k(d^{\text{LP}}(\nu^+)) \geq \epsilon_1[m_k(0) - m_k(d^{\text{LP}}(\nu_\infty))]$.
    Endif
Endif
Increase $\nu^+$ if necessary to satisfy
            $l_{\nu^+}(0) - l_{\nu^+}(d^{\text{LP}}(\nu^+)) \geq \epsilon_2 \nu^+[m_k(0) - m_k(d^{\text{LP}}(\nu^+))]$.
Set $\nu_k \leftarrow \nu^+$ and $d_k^{\text{LP}} \leftarrow d^{\text{LP}}(\nu^+)$.
```

The selection of $\nu^+ > \nu_{k-1}$ is achieved in all cases by successively increasing the current trial value of ν by 10 and re-solving the linear program. The penalty update algorithm above guarantees that ν is chosen large enough to ensure convergence to a stationary point [4]. Although the procedure does require the solution of some additional linear programs, our experience is that it results in an overall savings in iterations (and total LP solves) by achieving a better penalty parameter value more quickly, compared with rules which update the penalty parameter based on monitoring progress in feasibility. In addition, the extra LP solves are typically very inexpensive requiring relatively few simplex iterations because of the effectiveness of warm starts when re-solving the LP with a different penalty parameter value.

5 Projected CG Iteration

One of the main modules shared by the algorithms implemented in KNITRO, is a projected conjugate gradient iteration. The tangential subproblem (3.12) in the INTERIOR/CG algorithm and the EQP phase (4.6) of the ACTIVE algorithm both require the solution of an equality constrained quadratic program. We solve these problems using a projected conjugate gradient iteration [10, 20, 24, 26, 32], which is well suited for large problems and can handle the negative curvature case without the need for Hessian modifications. We now outline this iteration and refer the reader to [20] for a more detailed derivation.

Consider the quadratic program

$$\min_x \quad \frac{1}{2}x^T G x + h^T x \tag{5.9a}$$

$$\text{subject to} \qquad Ax = b, \tag{5.9b}$$

and assume that G is positive definite on the null space of A. One way to solve (5.9) is to eliminate the constraints (5.9*b*) and apply the conjugate gradient method to the reduced problem. An equivalent strategy is to apply a special form of the CG iteration to the KKT system of (5.9), which is given by

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} = \begin{bmatrix} -h \\ b \end{bmatrix}. \tag{5.10}$$

Although the coefficient matrix is not positive definite, we can apply the CG method to (5.10), provided we precondition and project the CG method so that it effectively solves the positive definite reduced problem within the feasible manifold (5.9*b*). This algorithm is specified below. Here we denote the preconditioning/projection operator by P and give its precise definition later on.

Algorithm PCG. Preconditioned Projected CG Method.
Choose an initial point x_0 satisfying $Ax_0 = b$. Set $x \leftarrow x_0$, compute $r = Gx + h$, $z = Pr$ and $p = -z$.
Repeat the following steps, until $\|z\|$ is smaller than a given tolerance:

$$\alpha = r^T z / p^T G p \tag{5.11}$$
$$x \leftarrow x + \alpha p \tag{5.12}$$
$$r^+ = r + \alpha G p \tag{5.13}$$
$$z^+ = P r^+ \tag{5.14}$$
$$\beta = (r^+)^T z^+ / r^T z \tag{5.15}$$
$$p \leftarrow -z^+ + \beta p. \tag{5.16}$$
$$z \leftarrow z^+ \quad \text{and} \quad r \leftarrow r^+ \tag{5.17}$$

End

This iteration has exactly the same form as the (standard) preconditioned CG method for solving symmetric and positive definite systems; see e.g. [19]. The crucial difference is that normally P is a symmetric and positive definite matrix, whereas in our case it represents a projection and preconditioning matrix, which we define (indirectly) as follows. Given a vector r, we compute $z = Pr$ as the solution of the system

$$\begin{bmatrix} D & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} z \\ w \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}, \tag{5.18}$$

where D is a symmetric matrix that is positive definite on the null space of A, and w is an auxiliary vector. For (5.18) to be a practical preconditioning operation, D should be a sparse matrix, so that solving (5.18) is significantly less costly than solving (5.10).

By construction $z = Pr$ is in the null space of A, and so are all the search directions generated by Algorithm PCG. Since initially $Ax_0 = b$, all subsequent iterates x also satisfy the linear constraints. To view this iteration relative to the reduced CG method in which we eliminate the constraints (5.9*b*) and apply CG to a problem of dimension $n-l$, note that all iterates of Algorithm PCG may be expressed as $x = x_0 + Zu$, for some vector $u \in R^{n-l}$, and where the columns of the $n \times (n-l)$ matrix Z form a basis for the null space of A. In these null-space coordinates the solution of the quadratic program (5.9) is given by the vector u that solves

$$(Z^T G Z)u = Z^T(Gx_0 + h). \tag{5.19}$$

It can be shown (see e.g. [20]) that the iterates x generated by Algorithm PCG are given by $x = x_0 + Zu$, where u are the iterates of the preconditioned conjugate gradient method on the system (5.19), using the matrix $Z^T D Z$ as a preconditioner. Therefore, Algorithm PCG is a standard preconditioned CG iteration as long as G and D are positive definite on the null space of A.

There are two advantages of following the approach of Algorithm PCG over the reduced CG approach. First, there is no need to compute a null space basis and consequently no risk that ill-conditioning in Z will deteriorate the rate of convergence of the CG iteration. Moreover, in the Interior/CG algorithm we first scale the slack variables by (3.13), so that the matrix A in (5.9) has the form

$$\begin{bmatrix} A_E & 0 \\ A_I & -S \end{bmatrix}. \tag{5.20}$$

Therefore there is no ill conditioning caused by some slack variables approaching 0. The second benefit is that the projection matrix in (5.18) can also be used to compute the normal step and Lagrange multipliers; thus the extra cost of each of these computations is only one back solve involving the factors of this projection matrix.

In the Interior/CG and Active algorithms we solve quadratic programs of the form (5.9) subject to a trust region constraint $\|x\| \le \Delta$; in addition, G may not be positive definite on the null space of A. We adapt Algorithm PCG to this case by following Steihaug's approach: we terminate Algorithm PCG if the trust region is crossed or if negative curvature is encountered.

Knitro 5.0 sets $D = I$ in (5.18) so that the preconditioner removes only ill-conditioning associated with the constraint matrix A. (We have experimented with other choices of D and future releases of Knitro will include banded and incomplete Cholesky preconditioners.)

Algorithm PCG assumes that an initial feasible point x_0 is provided. The factorization of the system in (5.18) allows us to compute such a point by solving

$$\begin{bmatrix} D & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ x_0 \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix},$$

which is in fact the minimum-norm solution in the norm weighted by D.

6 Special Algorithmic Features

The Knitro package provides many algorithmic options and features that are listed comprehensibly in the documentation that accompanies the software [37]. Here we highlight some of these options and discuss their relationship to the algorithms presented in the previous sections.

Hessian Options

The user can supply first and second derivatives, which generally results in the greatest level of robustness and efficiency for the three algorithms in Knitro. In some applications, however, the Hessian of the Lagrangian $\nabla^2_{xx}\mathcal{L}$ cannot be computed or is too large to store, but products of this Hessian times vectors can be obtained through automatic differentiation tools, adjoint codes or user-provided routines. For this case the Interior/CG and Active algorithms allow the user to provide these Hessian vector products at every iteration of the projected CG iteration. In a related option, Knitro takes control of this process and approximates the Hessian-vector products by finite differences of gradients of the Lagrangian; in this case the user need only provide gradients of the objective and constraints.

Quasi-Newton options have also been implemented for the three algorithms in Knitro. Here, the Hessian of the Lagrangian $\nabla^2_{xx}\mathcal{L}$ is replaced by a quasi-Newton approximation B_k, which is updated by the BFGS, SR1 or limited memory BFGS formulae. For example, for the interior-point methods, we define

$$\Delta l = \nabla_x \mathcal{L}(x^+, s^+, y^+, z^+) - \nabla_x \mathcal{L}(x, s^+, y^+, z^+), \qquad \Delta x = x^+ - x,$$

and substitute the correction pairs $(\Delta l, \Delta x)$ in the standard definition of the BFGS, SR1 or limited memory BFGS update formulae (see e.g. [31]). To ensure positive definiteness of the BFGS and L-BFGS updates the vector Δl is modified, if necessary, using Powell's damping procedure. SR1 updating is safeguarded to avoid unboundedness, but is allowed to generate indefinite approximations.

Feasible Mode.

In some applications, it is desirable for all of the iterates generated by the optimization algorithm to be feasible with respect to some or all of the *inequality* constraints. For example, the objective function may be defined only when some of the constraints are satisfied, making this feature absolutely necessary. Interior-point methods provide a natural framework for deriving feasible algorithms, and we have therefore developed versions of the Interior/CG and Interior/Direct algorithms that have this feature.

The adaptation is simple. If the current iterate x satisfies $c_I(x) > 0$, then after computing the step d, we let $x^+ = x + d_x$, redefine the slacks as

$$s^+ \leftarrow c_I(x^+), \tag{6.21}$$

and test whether the point (x^+, s^+) is acceptable for the merit function ϕ_ν. If so, we define this point to be the new iterate; otherwise we reject the step d and compute a new, shorter, trial step (in a line search algorithm we backtrack, and in a trust-region method we compute a new step with a smaller trust region). This strategy is justified by the fact that, if at a trial point we have that $c_i(x^+) \leq 0$ for some inequality constraint, the value of the merit function is $+\infty$, and we reject the trial point. This strategy also rejects steps $x + d_x$ that are too close to the boundary of the feasible region because such steps increase the barrier term $-\mu \sum_{i=1}^m \log(s_i)$ in the merit function (3.3). Apart from the reset (6.21), in the INTERIOR/CG algorithm we must introduce a slight modification [8] in the normal step computation to ensure that this step makes sufficient progress toward feasibility.

Initial Point Strategy.

As is well known, interior methods can perform poorly if the initial point is unfavorable. To overcome this problem, we have implemented several initial point strategies that work well for linear and quadratic programming and are also appropriate for nonlinear programs. At present, the initial point strategies are available only in the INTERIOR/DIRECT option. We now describe one of these strategies.

We first compute, at the user supplied initial point x_0, an affine scaling step $d^A = (d_x^A, d_s^A, d_y^A, d_z^A)$ by setting $\mu = 0$ in (3.4). Then we define

$$s_1 = \max(1, |s_0 + d_s^A|), \quad z_1 = \max(1, |z_0 + d_z^A|),$$

where the max and absolute values are applied component-wise. The primal variables x and the equality-constraint multipliers y are not altered, i.e., we define $(x_1, y_1) = (x_0, y_0)$. Finally we define the initial value of the barrier parameter as $\mu_1 = s_1^T z_1/m$.

The motivation for this strategy is to take care that the initial slacks and inequality multipliers are not too close to the feasible boundary which can lead to slow progress, and ideally to generate an initial point nearer to the central path. Furthermore, nonlinear programming algorithms compute only local minimizers and accept user-supplied initial estimates x_0 that often lie in the vicinity of a minimizer of interest. Therefore initial point strategies should either respect the user-supplied estimate x_0 or compute one that is not too distant from it. In addition, large initial values of the multipliers should be avoided in the Hessian since they may introduce unnecessary non-convexities in the problem. In particular if one of the components, say z_1^i, is large and the corresponding Hessian term $\nabla^2 c_1(x_1)$ is indefinite, the Hessian of the Lagrangian can become indefinite, slowing down the iteration. Therefore, when computing the first step of the interior algorithm from (x_1, s_1, y_1, z_1) we evaluate the Hessian of the Lagrangian using z_0 and not z_1, i.e., $\nabla_{xx}^2 \mathcal{L}(x_0, s_0, y_0, z_0)$ (this Hessian is independent of s, so the choice of that variable is irrelevant). More details about the initial point strategies are given in [17].

Special Problem Classes

When the nonlinear program (1.1) has a special form, the algorithms in KNITRO often reduce to well-known special-purpose methods.

For unconstrained optimization problems, the INTERIOR/CG and ACTIVE algorithms (using second derivatives) reduce to an inexact Newton-CG method with trust regions. This is because, in the unconstrained case, these algorithms skip their respective first phases, and compute the step using, respectively, the tangential subproblem (3.12) and the EQP phase (4.6), which are identical in this case. In the INTERIOR/DIRECT option, the algorithm will attempt to compute the Cholesky factorization of the Hessian, and if it is positive definite a backtracking line search will be performed along the Newton direction. If the Hessian is not positive definite, the algorithm reverts to the trust region INTERIOR/CG algorithm and therefore computes an inexact Newton-CG step.

If the problem (1.1) is a system of nonlinear equations, the algorithms in KNITRO implement a form of Newton's method (if second derivatives are provided). In INTERIOR/CG, only the normal step (3.11) is computed, and the resulting algorithm coincides with the Levenberg-Marquardt trust region method. The INTERIOR/DIRECT algorithm reduces to a line search Newton method in this case, using as a merit function the Euclidean norm of the residuals of the system of equations. If the Jacobian is singular, INTERIOR/DIRECT reverts to the Levenberg-Marquardt method.

KNITRO adapts itself automatically to the two classes of problems just discussed (unconstrained minimization and nonlinear equations). If the problem is a linear or quadratic program, the user must inform KNITRO, so that the algorithms can take full advantage of this fact. For LPs or QPs, INTERIOR/DIRECT is the recommended interior-point option and automatically enables the initial point strategy described above, as well as a more aggressive barrier update strategy. ACTIVE reduces to a simplex method in the LP case.

Infeasibility detection

It is not rare for users to generate optimization problems that do not have a feasible solution, and KNITRO includes heuristics to attempt to diagnose this situation. As is well known, however, infeasibility detection is a very difficult problem for nonlinear constraints, and the algorithms in KNITRO cannot distinguish between infeasible problems and convergence to an (infeasible) stationary point for a measure of feasibility.

In the interior point algorithms, our heuristics are based on the theory developed in [3]. It states that, if the interior point algorithm is not capable of finding a feasible point, then we have that $A_E(x_k)^T c_E(x_k) \to 0$, and $A_I(x_k)^T {c_I}^-(x_k) \to 0$, where ${c_I}^- = \max(0, -c_I)$. The KNITRO interior-point algorithms will terminate if these vectors are sufficiently small while $\|(c_E(x_k), {c_I}^-(x_k))\|$ stays above some level.

Since the algorithm implemented in ACTIVE is a penalty method, it can deal naturally with infeasibility. If a problem is infeasible then the penalty parameter will be driven to infinity. Moreover, if the algorithm is converging to a stationary point for our infeasibility measure, we have

$$m_k(0) - m_k(d^{\mathrm{LP}}(\nu_\infty)) \to 0,$$

during the penalty update procedure providing a clear indication of local infeasibility.

7 Crossover

Interior methods provide only an approximate estimate of the solution and the optimal active set. In many practical applications, however, it is useful to know precisely which constraints are active because this corresponds to the presence or activity of certain constituents of the solution. In addition, it is often important to have accurate estimates of the Lagrange multipliers (or sensitivities). This can be done by switching from the interior to an active-set iteration, a process that is often called *crossover*. Although crossover techniques have received much attention in the context of linear programming [27], to the best of our knowledge, none of the nonlinear interior codes provide an option for it. We regard it as essential to have this facility in our integrated system, both for computational efficiency, and to return solutions in a form that is useful for applications.

In linear programming, crossover involves two stages: identifying active constraints, and moving from a nonbasic optimal solution to a nearby basic one. In nonlinear programming, of course, we cannot expect the set of active constraints to correspond to a basic solution. Instead, our crossover procedure seeks to identify a set of active constraints with linearly independent constraint gradients, and computes a solution at which those constraints are satisfied with near equality, and which satisfies Lagrangian stationarity using these constraints only.

This crossover procedure is implemented by internally switching to the Active algorithm after the Interior/Direct or Interior/CG algorithm has solved the problem to the requested tolerance. We first solve the EQP phase of Active using a tolerance-based active-set estimate, and minimize the model function (4.5) along the resulting step direction to generate a new solution estimate. If this step does not solve the problem immediately, we begin the full Active algorithm with an initial LP trust region based on that active-set estimate. The goal is to judiciously choose the initial LP trust region small enough to exclude all the inactive constraints, but large enough to include the active ones. Below is a basic description of the Knitro crossover procedure.

Algorithm Crossover. Knitro Crossover Procedure.

1. The interior-point iteration terminates with stopping tolerance ϵ_{TOL} at iterate (x_k, s_k, y_k, z_k).

2. Estimate the set of active constraints, $\mathcal{A}$, using a tolerance test based on primal-dual feasibility and complementarity.

3. Using this active-set estimate, generate a step by solving the EQP given by (4.6) for d^{Q} and perform a line search to compute the steplength α^{Q}. If $x_k + \alpha^{\text{Q}} d^{\text{Q}}$ satisfies the stopping tolerances, terminate with that value and the corresponding multipliers.

4. Otherwise determine the initial LP trust region Δ_0^{LP}, and penalty parameter ν_0 for the Knitro-Active algorithm (Algorithm 4.1):

$$\begin{aligned} \Delta_0^{\text{LP}} &= \min\left\{\frac{c_i(x_k, s_k)}{\|\nabla c_i(x_k, s_k)\|} : i \notin \mathcal{A}\right\}, && (7.22) \\ \nu_0 &= 10\|(y_k, z_k)\|_\infty. && (7.23) \end{aligned}$$

5. Start Knitro-Active using initial point (x_k, s_k, y_k, z_k), Δ_0^{LP} and ν_0.

Initially in Step 3 of crossover, the active set is estimated using a tolerance test rather than by solving the LP (4.3). This is because, on some difficult problems, the cost of solving the LP subproblem can be non-trivial and we would like the cost of our crossover procedure in most cases to be a small part of the overall solution time. Therefore, if it is not necessary to solve the LP to identify the optimal active set, we seek to avoid doing this. In many cases, especially if strict complementarity holds at the solution, the initial estimate of the active set based on the simple tolerance test will be correct and the crossover will succeed in one iteration without solving any LPs.

The condition (7.22) used to initialize the initial LP trust region Δ^{LP} guarantees that if our active-set estimate is correct, the initial LP trust region will be small enough to exclude all inactive constraints. Motivated by the theory for the ℓ_1 exact penalty function, the penalty parameter ν is initialized to be a little larger than the Lagrange multiplier of largest magnitude at the interior-point solution.

Acknowledgment. The authors are grateful to several collaborators who have contributed to the development of KNITRO. Todd Plantenga and Mary Beth Hribar developed the software modules that led to the first version of the INTERIOR/CG algorithm in KNITRO. Guanghui Liu, Marcelo Marazzi, José Luis Morales and Dominique Orban contributed algorithmic improvements and performed extensive testing. Nick Gould collaborated with us on the development of the SLQP algorithm and its convergence analysis, and Jean-Charles Gilbert studied with us the convergence properties of the INTERIOR/CG algorithm.

References

[1] E.D. Andersen and K.D. Andersen. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In H. Frenk, K. Roos, T. Terlaky, and S. Zhang, editors, *High Performance Optimization*, pages 197–232, Dordrecht, The Netherlands, 2000. Kluwer Academic Publishers.

[2] J. Betts, S. K. Eldersveld, P. D. Frank, and J. G. Lewis. An interior-point nonlinear programming algorithm for large scale optimization. Technical report MCT TECH-003, Mathematics and Computing Technology, The Boeing Company, P.O. Box 3707, Seattle WA 98124-2207, 2000.

[3] R. H. Byrd, J.-Ch. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89(1):149–185, 2000.

[4] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. On the convergence of successive linear-quadratic programming algorithms. Technical Report OTC 2002/5, Optimization Technology Center, Northwestern University, Evanston, IL, USA, 2002.

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

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

[7] R. H. Byrd, M. Marazzi, and J. Nocedal. On the convergence of Newton iterations to non-stationary points. *Mathematical Programming, Series A*, 99:127–148, 2004.

[8] R. H. Byrd, J. Nocedal, and R. A. Waltz. Feasible interior methods using slacks for nonlinear optimization. *Computational Optimization and Applications*, 26(1):35–61, 2003.

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

[10] T. F. Coleman. Linearly constrained optimization and projected preconditioned conjugate gradients. In J. Lewis, editor, *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, pages 118–122, Philadelphia, USA, 1994. SIAM.

[11] A. R. Conn, N. I. M. Gould, and Ph. Toint. *Trust-region methods.* MPS-SIAM Series on Optimization. SIAM publications, Philadelphia, Pennsylvania, USA, 2000.

[12] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for Large-scale Nonlinear Optimization (Release A).* Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York, 1992.

[13] E. D. Dolan, J. J. Moré, and T. S. Munson. Optimality measures for performance profiles. Technical report, Argonne National Laboratory, Argonne, IL, USA, 2004.

[14] A. Drud. CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31:153–191, 1985.

[15] R. Fletcher and S. Leyffer. User manual for filtersqp. Technical Report NA/181, Dundee, Scotland, 1998.

[16] R. Fletcher and E. Sainz de la Maza. Nonlinear programming and nonsmooth optimization by successive linear programming. *Mathematical Programming*, 43(3):235–256, 1989.

[17] E. Michael Gertz, J. Nocedal, and A. Sartenaer. A starting-point strategy for nonlinear interior methods. *Applied Math Letters*, 5, 2004.

[18] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 2002.

[19] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, second edition, 1989.

[20] N. I. M. Gould, M. E. Hribar, and J. Nocedal. On the solution of equality constrained quadratic problems arising in optimization. *SIAM Journal on Scientific Computing*, 23(4):1375–1394, 2001.

[21] Christelle Guéret, Christian Prins, and Marc Sevaux. *Applications of optimization with Xpress-MP.* Dash Optimization, 2002.

[22] C. Gurwitz and M. L. Overton. Sequential quadratic programming methods based on approximating a projected hessian matrix. *SIAM Journal on Scientific and Statistical Computing*, 10(4):631–653, 1989.

[23] ILOG CPLEX 8.0. *User's Manual.* ILOG SA, Gentilly, France, 2002.

[24] C. Keller, N. I. M. Gould, and A. J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1300–1317, 2000.

[25] M. Kocvara. `http://www2.am.uni-erlangen.de/~kocvara/pennon/ampl-nlp-pp.html`, 2003. Results of NLP problems: performance profiles.

[26] L. Lukšan and J. Vlček. Indefinitely preconditioned inexact Newton method for large sparse equality constrained nonlinear programming problems. *Numerical Linear Algebra with Applications*, 5(3):219–247, 1998.

[27] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in mathematical programming: Interior-point and related methods*, chapter 8, pages 131–158. Springer-Verlag, New York, 1989.

[28] J. L. Morales, J. Nocedal, R. A. Waltz, G. Liu, and J. P. Goux. Assessing the potential of interior methods for nonlinear optimization. In L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. van Bloemen Waanders, editors, *Large-Scale PDE-Constrained Optimization*, volume 30, pages 167–183, Heidelberg, Berlin, New York, 2003. Springer Verlag. Lecture Notes in Computational Science and Engineering.

[29] B. A. Murtagh and M. A. Saunders. MINOS 5.4 user's guide. Technical report, SOL 83-20R, Systems Optimization Laboratory, Stanford University, 1983. Revised 1995.

[30] J. Nocedal, A. Wächter, and R. A. Waltz. Adaptive barrier strategies for nonlinear interior methods. Technical Report RC 23563, IBM Watson Research Center, Yorktown Heights, NY, USA, 2005.

[31] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Series in Operations Research. Springer, 1999.

[32] B. T. Polyak. The conjugate gradient method in extremal problems. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 9:94–112, 1969.

[33] R. J. Vanderbei and D. F. Shanno. An interior point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[34] A. Wächter. *An interior point algorithm for large-scale nonlinear optimization with applications in process engineering.* PhD thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, 2002.

[35] A. Wächter and L. T. Biegler. Failure of global convergence for a class of interior point methods for nonlinear programming. *Mathematical Programming*, 88(3):565–574, 2000.

[36] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Technical Report RC 23149, IBM T. J. Watson Research Center, Yorktown Heights, NY, USA, March 2004.

[37] R. A. Waltz. KNITRO 4.0 User's Manual. Technical report, Ziena Optimization, Inc., Evanston, IL, USA, October 2004.

[38] R. A. Waltz, J. L. Morales, J. Nocedal, and D. Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. Technical Report 2003-6, Optimization Technology Center, Northwestern University, Evanston, IL, USA, June 2003. To appear in Mathematical Programming A.