AN AUGMENTED LAGRANGIAN INTERIOR-POINT METHOD USING DIRECTIONS OF NEGATIVE CURVATURE

Javier M. Moguerza
and Francisco J. Prieto

00-36

WORKING PAPERS

Universidad Carlos III de Madrid

Working Paper 00-36
Economics Series 14
May 2000

Departamento de Estadística y Econometría
Universidad Carlos III de Madrid
Calle Madrid, 126
28903 Getafe (Spain)
Fax (34) 91 624-98-49

AN AUGMENTED LAGRANGIAN INTERIOR-POINT METHOD USING DIRECTIONS OF NEGATIVE CURVATURE

Javier M. Moguerza and Francisco J. Prieto *

Abstract

We describe an efficient implementation of an interior-point algorithm for non-convex problems that uses directions of negative curvature. These directions should ensure convergence to second-order KKT points and improve the computational efficiency of the procedure. Some relevant aspects of the implementation are the strategy to combine a direction of negative curvature and a modified Newton direction, and the conditions to ensure feasibility of the iterates with respect to the simple bounds. The use of multivariate barrier and penalty parameters is also discussed, as well as the update rules for these parameters. Finally, numerical results on a set of test problems are presented.

Keywords: Primal-dual methods; nonconvex optimization; line searches

* Moguerza, Dept. of Statistics and Econometrics, Univ. Carlos III de Madrid, Spain, E-mail: moguerza@est-econ.uc3m.es; Prieto, Dept. of Statistics and Econometrics, Univ. Carlos III de Madrid, Spain. E-mail: fjp@est-econ.uc3m.es. AMS: 49M37, 65K05, 90C30

An augmented Lagrangian interior-point method using directions of negative curvature

Javier M. Moguerza[1] Francisco J. Prieto[1]
Dept. of Statistics and Econometrics
Univ. Carlos III de Madrid, Spain
E-mail: moguerza@est-econ.uc3m.es E-mail: fjp@est-econ.uc3m.es

ABSTRACT

We describe an efficient implementation of an interior-point algorithm for non-convex problems that uses directions of negative curvature. These directions should ensure convergence to second-order KKT points and improve the computational efficiency of the procedure. Some relevant aspects of the implementation are the strategy to combine a direction of negative curvature and a modified Newton direction, and the conditions to ensure feasibility of the iterates with respect to the simple bounds. The use of multivariate barrier and penalty parameters is also discussed, as well as the update rules for these parameters. Finally, numerical results on a set of test problems are presented.

Keywords: Primal-dual methods; Nonconvex optimization; Line searches

AMS: 49M37, 65K05, 90C30

1 Introduction

We are interested in developing an algorithm to compute local solutions for nonlinear, and possibly non-convex, problems of the form

$$\begin{array}{ll} \min_x & f(x) \\ \text{s.t.} & c(x) = 0 \\ & x \geq 0, \end{array} \tag{1}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $c : \mathbb{R}^n \mapsto \mathbb{R}^m$. More specifically, we wish to compute second-order KKT points for problem (1), that is, points that can be assured to satisfy both the first-order necessary conditions and the second-order condition.

The use of directions of negative curvature plays a crucial role in this context; only by using this second-order information it is possible to ensure convergence to such points. Trust-region methods are able to take negative curvature implicitly into account when computing the update direction, if exact second derivatives are used to form the corresponding subproblems. Line search procedures, while presenting interesting properties from a practical point of view, must take second-order information into account by explicitly computing some direction of negative curvature, if it is available. The idea of using directions of negative curvature was proposed by Fiacco and McCormick [5]. Later, Moré and Sorensen [15] described how to modify Newton's method using these ideas.

The explicit computation of these directions can be carried out with limited cost, by taking advantage of a factorization of the coefficient matrix in the system of Newton equations, if exact second derivatives are used, see [10] for example. Nevertheless, the requirement to obtain both descent and negative curvature directions from the Newton system of equations limits the choice of numerical procedures that can be used to compute the search direction.

In this paper we will be concerned with deriving a line search algorithm that uses negative curvature. The directions required to update the iterates will be generated using an interior-point approach. In this setting, problem (1) is transformed into a sequence of equality constrained problems of the form (see [5])

$$\begin{array}{ll} \min_x & f(x) - \sum_i \mu_i \log x_i \\ \text{s.t.} & c(x) = 0. \end{array} \tag{2}$$

[1]This research was supported by DGICYT grant PB96-0111.

The search directions are computed to approximate the solutions of these barrier problems. For computational efficiency reasons we have chosen to use a vector of barrier parameters $\mu \in \mathbb{R}^n$, one for each simple bound $x \geq 0$.

Interior-point methods have proved to be very successful for the solution of linear and general convex problems. More recently, a significant amount of effort has been devoted to extending these procedures to non-convex problems. Among the several proposals in the literature, we could mention those of El-Bakry et al. [3], Gajulapalli [7], Gay et al. [8], Vanderbei and Shanno [18], Yamashita and Yabe [23], etc. Nevertheless, very few of these proposals have taken into consideration the use of negative curvature directions.

Once negative curvature directions have been obtained, it is still necessary to combine them with traditional descent directions. Iterates in general will fail to satisfy both first-order and second-order conditions, and it is important to make use of both types of information simultaneously to ensure the efficiency of the procedure. The combination of descent and negative curvature directions has been considered by McCormick [13], Moré and Sorensen [15] or Forsgren and Murray [6], among others. Designing an efficient procedure to obtain a satisfactory combination of these directions poses considerable difficulties, as the Newton direction is well-scaled in general, and particularly so near a stationary point, while directions of negative curvature have no inherent scale.

A line search has been introduced in our algorithm as a mechanism to ensure global convergence. We will compute the iterates in such a manner that the value of an augmented Lagrangian merit function is decreased in each iteration. For problem (2) this merit function takes the form

$$LA(x, \lambda; \rho) = f(x) - \sum_i \mu_i \log x_i - \lambda^T c(x) + \frac{1}{2} \sum_j \rho_j c_j(x)^2. \tag{3}$$

The penalty term in the merit function makes use of a vector of penalty parameters, one for each constraint, $\rho \in \mathbb{R}^m$. This function has been extensively studied by Bertsekas [1] among others. It has the advantage of being differentiable at all points where it is defined (the interior of the positive orthant). Also, under suitable assumptions the local minimizers for problem (2) are minimizers for this merit function, if all components of ρ are large enough.

This merit function introduces parameters and variables, λ and ρ, that have to be updated through successive iterations to ensure convergence. Analogously, the barrier problems (2) also include parameters μ that must be updated. The choice of updating strategies may affect significantly the efficiency of the overall procedure and will be considered in some detail in the following sections.

The aim of this paper is to address the three issues discussed in the preceding paragraphs: the definition of the search directions, their combination and the updating of the parameters, in a manner that produces an efficient and robust procedure. We are also interested in determining from computational experiments the impact that using negative curvature information may have on the efficiency of the algorithm. In this regard, we will present and discuss some computational results on a test problem set.

The paper is organized as follows: In Section 2 we describe the procedure to compute the search directions in the algorithm. In Section 3 we indicate how to combine the directions to obtain the next iterate. In Section 4 we present the rules for updating the algorithm parameters. In Section 5 we discuss some implementation issues and list the general structure of the algorithm. Finally, in Section 6 we present and comment some computational results on a set of small test problems.

2 Computation of the search directions

2.1 Definitions and notation

The first-order Karush-Kuhn-Tucker (KKT) conditions for problem (1) are:

$$\begin{aligned} \nabla_x f(x) - \nabla_x c^T(x)\lambda - \sigma &= 0, \\ c(x) &= 0, \\ \Sigma x &= 0, \\ x, \sigma &\geq 0, \end{aligned} \tag{4}$$

where Σ denotes a diagonal matrix having as entries the elements of σ, $\Sigma = \text{diag}(\sigma)$.

In the proposed algorithm, instead of considering directly the preceding conditions, we solve a sequence of problems (2) such that $\mu_i \to 0$ for all i, following [5]. The first-order KKT conditions for (2) are:

$$\begin{aligned} \nabla_x f(x) - \nabla_x c^T(x)\lambda - X^{-1}\mu &= 0, \\ c(x) &= 0, \end{aligned} \tag{5}$$

where $X = \text{diag}(x)$. Replacing

$$\sigma = X^{-1}\mu, \tag{6}$$

in the first equation of (5), the first-order KKT conditions for the barrier problem can be rewritten as:

$$\begin{aligned} \nabla_x f(x) - \nabla c^T(x)\lambda - \sigma &= 0, \\ c(x) &= 0, \\ \Sigma x &= \mu. \end{aligned} \tag{7}$$

The set of equations (7) is known as the primal-dual equations for problem (2). Initially implemented by Mehrotra [14], the search directions obtained from them have been shown in practice to be computationally more efficient than those computed from (5). The algorithm will compute search directions based on these primal-dual KKT equations. In addition to trying to satisfy these conditions, the algorithm should also ensure that the variables x_i remain strictly positive, as the logarithmic terms in the objective function of (2) are only defined on this set. From the comparison of these conditions and (4), we also need to ensure that $\sigma \geq 0$ in all iterations of the algorithm.

We will also want to satisfy the necessary second-order condition. For problem (2) this condition requires

$$Z_A^T \left(\nabla^2_{xx} L(x,\lambda) + MX^{-2} \right) Z_A \quad \text{p.s.d.}, \tag{8}$$

where L denotes the Lagrangian function for (2), $L(x,\lambda) = f(x) - \sum_i \mu_i \log x_i - \lambda^T c(x)$, M is a diagonal matrix with entries those of μ, $M = \text{diag}(\mu)$ and Z_A has columns that form a basis for the null-space of $\nabla c(x)$.

The algorithm we propose must carry out the following tasks in each iteration:

- The computation of search directions, both to improve the satisfaction of the first-order KKT conditions (7) using Newton's method, and the satisfaction of the second-order condition (8) using directions of negative curvature.
- The combination of these directions to compute the next iterate, ensuring sufficient descent for the augmented Lagrangian merit function (3).
- Finally, the updating of the multipliers λ and penalty parameters ρ in the merit function, and the barrier parameters μ associated with problem (2).

In the following paragraphs we indicate how to conduct these tasks in an efficient manner. We start by considering the computation of a descent direction for the variables x, d_x, based on a modified Newton method applied to the primal-dual equations (7).

2.2 The descent direction

Newton's method provides search directions d_x, d_λ and d_σ, corresponding to update directions for the variables x, λ and σ respectively. From the first-order Taylor series expansion for the primal-dual KKT conditions (7) about the values x, λ and σ, the resulting system of linear equations defining the search directions is:

$$\begin{pmatrix} H & -\nabla_x c^T & -I \\ \nabla_x c & & \\ & \Sigma & X \end{pmatrix} \begin{pmatrix} d_x \\ d_\lambda \\ d_\sigma \end{pmatrix} = \begin{pmatrix} -\nabla_x f + \nabla_x c^T \lambda + \sigma \\ -c \\ \mu - \Sigma x \end{pmatrix}, \tag{9}$$

where $H = \nabla^2_{xx} L(x, \lambda)$, $\Sigma = \text{diag}(\sigma)$, I denotes the identity matrix and $X = \text{diag}(x)$.

From the last set of equations in (9), we have

$$d_\sigma = X^{-1}\mu - \sigma - X^{-1}\Sigma d_x. \tag{10}$$

Substituting (10) in the first two sets of equations in (9), the movement direction d_x can be computed as the solution of the symmetric system

$$K \begin{pmatrix} d_x \\ -d_\lambda \end{pmatrix} = \begin{pmatrix} -\nabla_x f + \nabla_x c^T \lambda + X^{-1}\mu \\ -c \end{pmatrix}, \tag{11}$$

where K is defined as

$$K = \begin{pmatrix} G & \nabla_x c^T \\ \nabla_x c & 0 \end{pmatrix}, \tag{12}$$

for $G = H + X^{-1}\Sigma$.

2.3 Solving the system of equations

The direction obtained from (11) may fail to provide descent for any reasonable merit function, for example when the iterates are close to a stationary point that is not a minimizer. To ensure good global convergence properties for the algorithm it is important that the direction d_x should provide sufficient descent for the merit function (3), and this requires adapting the solution of system (11).

The gradient of the merit function is given by

$$\nabla_x LA = \nabla_x f - X^{-1}\mu - \nabla_x c^T(\lambda - Rc), \tag{13}$$

where $R = \text{diag}(\rho)$. The Hessian of LA with respect to x will be given by

$$\begin{aligned} \nabla^2_{xx} LA &= \nabla^2_{xx} f + MX^{-2} - \textstyle\sum_j (\lambda_j - \rho_j c_j)\nabla^2_{xx} c_j + \nabla_x c^T R \nabla_x c \\ &= \nabla^2_{xx} L(x, \lambda - Rc) + MX^{-2} + \nabla_x c^T R \nabla_x c. \end{aligned} \tag{14}$$

Considering the Newton direction for the merit function (3), defined as $\nabla^2_{xx} LA d_x = -\nabla_x LA$, or equivalently

$$(\nabla^2_{xx} L(x, \lambda - Rc) + MX^{-2} + \nabla_x c^T R \nabla_x c) d_x = -(\nabla_x f - X^{-1}\mu - \nabla_x c^T(\lambda - Rc)). \tag{15}$$

As the merit function is not very efficient at ensuring the satisfaction of the constraints, we impose the additional condition that $\nabla_x c(x)\bar{d}_x = -c(x)$, already included in (11). Condition (15) now becomes

$$(\nabla^2_{xx} L(x, \lambda - Rc) + MX^{-2}) d_x = -(\nabla_x f - X^{-1}\mu - \nabla_x c^T \lambda). \tag{16}$$

To ensure that d_x is a direction of descent for (3), we may replace the coefficient matrix $G_\rho = \nabla^2_{xx} L(x, \lambda - Rc) + MX^{-2}$ in the preceding condition with a matrix $\bar{G}_\rho = \bar{H} + X^{-1}\Sigma$ that is positive definite on the null space of the constraints $\nabla_x c(x)$, that is, $\bar{G}_\rho$ should be such that $Z_A^T \bar{G}_\rho Z_A$ is positive definite.

The modified system used to define the search directions is then

$$\begin{pmatrix} \bar{G}_\rho & \nabla_x c^T \\ \nabla_x c & 0 \end{pmatrix} \begin{pmatrix} d_x \\ -d_\lambda \end{pmatrix} = \begin{pmatrix} -\nabla_x f + \nabla_x c^T(\lambda - Rc) + X^{-1}\mu \\ -c \end{pmatrix}, \tag{17}$$

where the coefficient matrix is computed from a modification of

$$K_\rho = \begin{pmatrix} G_\rho & \nabla_x c^T \\ \nabla_x c & 0 \end{pmatrix}, \tag{18}$$

for $G_\rho = \nabla^2_{xx} L(x, \lambda - Rc) + X^{-1}\Sigma$.

The matrix $\bar{G}_\rho$ in (17) can be generated in the process of factorizing K_ρ. A modified Choleski factorization of the reduced Hessian $Z_A^T G_\rho Z_A$ could be used, as in [8]. This approach requires forming

explicitly the reduced Hessian, and as a consequence it is limited to problems in which this reduced Hessian is small. We have chosen to use a version of the symmetric indefinite factorization, see [2] for example, incorporating the modifications proposed in [6]. This alternative is able to obtain the desired modification for the reduced Hessian directly from system (17), it allows the computation of appropriate directions of negative curvature, as we will describe in Section 4, and it can be applied to medium-sized and large problems.

The details of the factorization used in the algorithm can be found in [6], but we now state its basic result. Assume that the LDL^T factorization of K_ρ, defined in (18), has been computed using the factorization algorithm described in [6], and that the matrix D in the factorization is partitioned into

$$D = \begin{pmatrix} D_1 & 0 \\ 0 & D_2 \end{pmatrix}, \tag{19}$$

where D_1 and D_2 are block-diagonal matrices with 1×1 and 2×2 blocks, D_1 includes all the pivots suitably chosen from elements of $\nabla c(x)$ and $D_2 = U\Lambda U^T$. The precise rules to choose these pivots can be found in [6]. Also, for a given $\varepsilon > 0$, define a diagonal matrix $\bar{\Lambda}$ having as its i-th diagonal element the value $\bar{\lambda}_i = \max(|\lambda_i|, \varepsilon)$. Construct $\bar{D}_2$ as $\bar{D}_2 = U\bar{\Lambda}U^T$ and define $\bar{D}$ as:

$$\bar{D} = \begin{pmatrix} D_1 & 0 \\ 0 & \bar{D}_2 \end{pmatrix}.$$

Let $\tilde{d}$ be the solution of

$$L\bar{D}L^T\tilde{d} = P^T b, \tag{20}$$

for P an appropriate permutation matrix, and b the right-hand side of (17). Finally let $d = P\tilde{d}$, the vector of unknowns in system (17),

$$d = \begin{pmatrix} d_x \\ -d_\lambda \end{pmatrix}.$$

It holds that $Ad_x = -c$ and $Z_A^T \bar{G}_\rho Z_A$ is positive definite, where $\bar{G}_\rho$ is the submatrix corresponding to the appropriate rows and columns of $L\bar{D}L^T$.

The factorization in [6] requires that the matrix $\nabla_x c$ should have full row rank. This cannot be guaranteed in practice, but our algorithm detects this rank deficiency within the factorization procedure and takes into account those rows of $\nabla_x c$ that seem to be linearly independent. This modification introduces errors in the solution of the system, but seems to behave reasonably well in practice.

2.4 Second-order directions

If we wish to avoid convergence to points not satisfying the second-order necessary condition (8), we must make use of negative curvature directions. For an unconstrained problem $\min_x f(x)$, we will look for directions satisfying the classical definition, see [15], that is, we will require a direction of negative curvature d at an iterate x to satisfy

$$\nabla f(x)^T d \leq 0 \qquad \text{and} \qquad d^T \nabla^2 f(x) d < 0. \tag{21}$$

For equality constrained problems these conditions can be easily generalized: as negative curvature information is only relevant on the subspace spanned by Z_A, see (8), we consider only those negative curvature directions that lie in this subspace, $d = Z_A v$. However, for nonlinearly constrained problems it is not clear how to define these directions, or how to use them, as now negative information will depend on the current estimate of the active set, and it may also be relevant when moving away from active constraints. We will follow the proposal by Murray and Prieto in [16], by building directions of negative curvature for the merit function based on the preceding unconstrained conditions, under certain conditions on the infeasibility of the current iterate.

We now give a more precise statement of the conditions we will impose. The gradient of the merit function (3) and its Hessian matrix with respect to x are given by (13) and (14), respectively. Analogously

to the case of the descent direction, and due to the limitations of the merit function regarding the satisfaction of the constraints, we will impose the additional condition that the direction of negative curvature should lie in the null-space of the matrix $\nabla_x c$, a reasonable requirement given that these constraints must hold with equality at the solution, and the descent direction is computed to satisfy these constraints. From the definition (21) and this additional condition, a direction of negative curvature d_n for our algorithm should satisfy

$$d_n = Z_A w, \qquad d_n^T(\nabla_x f - X^{-1}\mu) \leq 0 \quad \text{and} \quad d_n^T(\nabla_{xx}^2 L(x, \lambda - Rc) + MX^{-2})d_n < 0. \tag{22}$$

At each iterate we need to determine if negative curvature is present, and if that is the case we also need to compute a direction satisfying the preceding conditions. We would conduct this analysis and compute this direction from system (11), used to define the descent direction, to reduce as much as possible the computational cost within the algorithm. However, the Hessian matrix that appears in that system has the form $H + X^{-1}\Sigma$, and it may differ from $H + MX^{-2}$, the matrix used in the definition (22). Both matrices will be close if (6) is approximately satisfied. As a consequence, we will compute the direction of negative curvature from (11), but we will check if conditions (22) are satisfied before using it in the search. Note that close to a stationary point for the barrier problem (2), condition (6) will be approximately satisfied and directions of negative curvature, if they exist, will eventually be accepted.

Another theoretical issue must be considered. The definition (22) has been made for the barrier problem (2), but the problem of interest is (1). It would be important to ensure that by computing second-order KKT points for problem (2), we are in fact solving problem (1). This issue has been treated in detail in [19] and [8]; we now present the basic result that justifies the validity of our approach.

A second-order KKT point (x^*, λ^*) for problem (1) will satisfy the first-order necessary conditions (4), and the matrix $Z_J^T \nabla^2 L(x^*, \lambda^*) Z_J$ will be positive semidefinite. Here, Z_J denotes a matrix whose columns form a basis for the null-space of the Jacobian of the active constraints at the solution of (1). If $\hat{I} \in \mathbb{R}^{p \times n}$ denotes the p rows of the identity associated with the components of x^* that are equal to zero (the active constraints in $x \geq 0$), then Z_J corresponds to a basis for the null-space of the matrix

$$\begin{pmatrix} \nabla c(x^*) \\ \hat{I} \end{pmatrix}.$$

For Z_A, a basis for the null-space of $\nabla c(x^*)$, we can always assume that Z_A and Z_J are chosen so that $Z_A = \begin{pmatrix} R & Z_J \end{pmatrix}$. Let $v^* = (x^*, \lambda^*, \sigma^*)$ denote the corresponding values at a second-order KKT point of (1). We will use the notation $H_\rho = \nabla^2 L(x, \lambda - Rc)$ and $G_\rho = H_\rho + X^{-1}\Sigma$. In Theorem 3.3 of [20] and Theorem V.2.7 of [17] it is shown that given $v = (x, \lambda, \sigma)$ such that $\|v - v^*\| < \epsilon$ for some small enough ϵ, for the matrix $Z_A^T G_\rho Z_A$ evaluated at v and $Z_A = \begin{pmatrix} R & Z_J \end{pmatrix}$, as indicated above, it holds that

- The largest (in magnitude) p eigenvalues of $Z_A^T G_\rho Z_A$ are positive and unbounded as $v \to v^*$.
- The remaining $n - m - p$ eigenvalues are within $O(\epsilon)$ of the eigenvalues of $Z_J^T H_\rho Z_J$.
- G_ρ has invariant subspaces for which there exist bases $\tilde{R}$ and $\tilde{Z}$ such that

$$\|R - \tilde{R}\| = O(\epsilon) \qquad \|Z_J - \tilde{Z}\| = O(\epsilon),$$

and in particular

$$\|\tilde{Z}^T G_\rho \tilde{Z} - Z_J^T H_\rho Z_J\| = O(\epsilon).$$

As a consequence of this result, the information of interest to our algorithm regarding the eigenvalues of $Z_J^T H_\rho Z_J$, the ones entering the definition of second-order KKT points for problem (1), can be derived by observing the finite eigenvalues of $Z_A^T G_\rho Z_A$ from system (11). Moreover, if we are close enough to a second-order KKT point of (1), the (finite) negative eigenvalues of $Z_A^T G_\rho Z_A$ and their associated eigenvectors will provide good approximations to the corresponding ones in $Z_J^T H_\rho Z_J$. As a consequence, we will be able to compute directions of negative curvature from (11) in an efficient manner, while ensuring convergence to second-order KKT points of (1).

2.5 Computation of directions of negative curvature

We compute a direction of negative curvature d_n (assuming that it exists) from the same symmetric indefinite factorization used to obtain the descent direction d_x in (20). If no negative curvature is available at the current iterate, we set $d_n = 0$. Let K_ρ be the matrix defined in (18), and assume that its symmetric indefinite factorization $K_\rho = LDL^T$ has been computed using the algorithm in [6]. Assume that from the factorization it has been determined that this matrix has more than m negative eigenvalues, implying that $Z_A^T G_\rho Z_A$ has at least one negative eigenvalue.

We will compute d_n in the following manner: Let w be defined as $w = P\tilde{w}$, where P is the permutation matrix in (20) and $\tilde{w}$ satisfies

$$\begin{pmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{pmatrix} \begin{pmatrix} \tilde{w}_1 \\ \tilde{w}_2 \end{pmatrix} = \pm\sqrt{-\lambda_{\min}(D_2)} \begin{pmatrix} 0 \\ u_\lambda \end{pmatrix}, \tag{23}$$

where $\lambda_{\min}(D_2)$ denotes the most negative eigenvalue of D_2, defined in (19), and u_λ is a unit eigenvector corresponding to this smallest eigenvalue. The negative curvature direction d_n is defined as the first n components of w. In [6] it is shown that $\nabla c(x) d_n = 0$, so that d_n lies in the correct subspace, and there exist positive constants c_1 and c_2 such that

$$d_n^T G d_n \leq -c_1 \lambda_{\min}^2(Z_A^T G Z_A) \qquad \text{and} \qquad d_n^T d_n \leq -c_2 \lambda_{\min}(Z_A^T G Z_A).$$

Nevertheless, this scaling may not be adequate for the search procedure. We rescale this negative curvature direction using the norm of the descent direction d_x, to ensure that both directions are comparable in size.

If condition (6) is approximately satisfied, the direction d_n computed using the preceding procedure will be a direction of negative curvature for the merit function $LA(x, \lambda; \rho)$. In this case, to satisfy (22) it will be enough to choose the sign of d_n so that

$$d_n^T(\nabla_x f - X^{-1}\mu) \leq 0. \tag{24}$$

As in general (6) may not be satisfied, each time a direction of negative curvature is computed we will also check if

$$d_n^T(H_\rho + MX^{-2})d_n < 0$$

is satisfied. If this is not the case, the negative curvature direction d_n will not be used.

3 The curvilinear search

For a given iterate (x, λ), classical line search methods applied to problem (2) compute a direction of movement

$$d = \begin{pmatrix} d_x \\ d_\lambda \end{pmatrix},$$

and then determine a scalar α ensuring that the next iterate $(x + \alpha d_x, \lambda + \alpha d_\lambda)$ provides sufficient decrease for an appropriate merit function. The role of the merit function is to ensure, through the proper choice of α, the convergence to a minimizer of this merit function, that should correspond to a minimizer of (2). The computation of α is usually referred to as a line search (see [10], for example).

This approach works quite well in practice whenever there is a single search direction d_x. In our case we may have a pair of search directions at a given iteration, d_x and d_n. In this case, the preceding procedure must be modified to take into account that the next iterate must be found by searching on a subspace of dimension two, instead of dimension one as was the case for the classical approach. We have chosen to use a curvilinear search, defined on the subspace generated by both directions. This curvilinear search will be applied to the augmented Lagrangian merit function (3).

This curvilinear search will be applied to the directions d_x computed from (17) and d_n obtained as described in the preceding section from (23). The search will also be carried out on the multipliers. Their

search direction will be defined from d_λ, obtained from (17), but it will be modified to take into account the right-hand side used in (17) and the Newton condition for the merit function given in (16). The actual search direction is defined as

$$d_{\lambda\rho} = d_\lambda - Rc(x). \tag{25}$$

To combine the preceding directions we will follow the proposals in [15] and [16]. Given an iterate (x, λ) and directions d_x, d_n and d_λ, the next iterate will be obtained as a point on the curves

$$x(\alpha) = x + \alpha^2 d_x + \alpha d_n, \tag{26}$$
$$\lambda(\alpha) = \lambda + \alpha^2 d_{\lambda\rho}. \tag{27}$$

The value of α is determined to ensure $(x(\alpha), \lambda(\alpha))$ provides sufficient descent for the augmented Lagrangian merit function (3). Let

$$\phi(\alpha) = LA(x(\alpha), \lambda(\alpha); \rho),$$

where ρ is the penalty parameter vector. For an initial value $\alpha_{\max}$, defined later on, we will check if

$$\phi(\alpha_{\max}) \leq \phi(0) + \gamma \frac{1}{2} \alpha_{\max}^2 \phi_k''(0).$$

If this condition is satisfied, we choose $\tilde{\alpha} = \alpha_{\max}$. Otherwise, we apply a backtracking procedure from $\alpha_{\max}$ to find a value $\tilde{\alpha} \in (0, \alpha_{\max})$ satisfying

$$\phi(\tilde{\alpha}) \leq \phi(0) + \gamma \frac{\tilde{\alpha}^2}{2} \phi''(0), \tag{28}$$
$$\phi'(\tilde{\alpha}) \geq \eta(\phi'(0) + \tilde{\alpha}\phi''(0)), \tag{29}$$

where γ and η are scalar parameters satisfying $0 < \gamma < \frac{1}{2}$ and $\frac{1}{2} < \eta < 1$.

In addition to the sufficient descent conditions (28)–(29), we also force the iterates to remain on the set where the merit function is defined, that is, α is chosen so that $x(\alpha) > 0$ by defining $\alpha_{\max}$ appropriately, that is, we choose $\alpha_{\max}$ so that all $\alpha \in [0, \alpha_{\max}]$ satisfy

$$P_i(\alpha) = \alpha^2 (d_x)_i + \alpha (d_n)_i + x_i > 0 \quad \forall i. \tag{30}$$

If $\bar{\alpha}_i$ denotes the smallest positive root of $P_i(\alpha)$, if it exists, or ∞ otherwise, the preceding condition is satisfied for a given i and all $\alpha \in [0, \bar{\alpha}_i)$, as $x_i > 0$. As a consequence, we must choose $\alpha_{\max} < \min_i \bar{\alpha}_i$.

On the other hand, to take advantage of the good local convergence properties of the Newton direction we would like to accept the step to this Newton direction ($\alpha = 1$) whenever it is reasonable, that is, whenever the Newton step lies within the positive orthant and there is no negative curvature available. Consequently, the initial step $\alpha_{\max}$ is defined as

$$\alpha_{\max} = \min(\delta \min_i(\bar{\alpha}_i), 1), \tag{31}$$

where the parameter δ, introduced to ensure the strict positivity of the iterates, has been defined as

$$\delta = \max(0.995, 1 - \|\mu\|_2). \tag{32}$$

Near the solution (when $\mu \simeq 0$), the term $1 - \|\mu\|_2$ guarantees that a step of one will not be prevented by the positivity requirement. This is relevant to ensure adequate local convergence properties.

Finally, we impose an additional condition that helps to ensure that the iterates remain in a compact set throughout the algorithm. The next iterate is computed as $x(\alpha^p)$, where $\alpha^p = \tilde{\alpha}$ if $\tilde{\alpha}$, satisfying (28)–(29), also satisfies

$$\|c(x(\alpha))\| \leq \beta_c. \tag{33}$$

Otherwise, a value α^p is determined by applying a backtracking procedure from $\tilde{\alpha}$ until conditions (28) and (33) are both satisfied.

4 Parameter updates

A complete specification of the algorithm should indicate how to update the different parameters that appear in the computation of the search directions and the curvilinear search. In the following paragraphs we specify the procedures used to update the multiplier estimates, the penalty parameters and the barrier parameters.

4.1 The multipliers

Two sets of dual variables are generated by the algorithm, the equality constraint multipliers λ and the approximations to the multipliers for the bound constraints σ. The multipliers λ are updated within the curvilinear search using (27) and the value α^p chosen for the variables x according to the procedure described in the preceding section.

The solution of Newtons system of equations (9) provides a search direction for the multipliers σ, d_σ, defined as (10). These dual variables will be updated from

$$\sigma(\alpha^d) = \sigma + \alpha^d d_\sigma,$$

using an adequate value of α^d. The only condition on the values of the dual variables is their non-negativity. The scalar α^d is chosen as the largest reasonable value that satisfies this condition, as follows. Let

$$\bar{\alpha}^d = \min\left(\delta \min\left(\frac{-\sigma_i}{(d_\sigma)_i} | (d_\sigma)_i < 0\right), 1\right),$$

where δ is defined as in (32). For $\vartheta = (\alpha^p/\alpha_{\max})^2$, α^d is defined as

$$\alpha^d = \vartheta \bar{\alpha}^d. \tag{34}$$

The value ϑ is introduced to scale α^d so that its value is related to the value of α^p obtained from the line search.

4.2 The penalty parameters

The penalty parameters ρ_j are used in the algorithm to ensure the convergence to points satisfying the constraints $c(x) = 0$. The Newton direction should provide iterates able to satisfy this condition, but if the penalty parameters are not sufficiently large, this Newton direction may not be a descent direction for the merit function and will not be accepted. As a consequence, the penalty parameters are chosen so that the sufficient decrease condition given by inequality (28) can be satisfied. The updating of these parameters is relevant in other ways related to the computational efficiency of the procedure. A very large value of these parameters may cause numerical problems in the computation of the search directions from (17). Also, these parameters have an impact on the updating of the λ multiplier estimates.

The update formula will be derived in terms of condition (28). This condition includes the initial derivatives of the merit function along the curve $x(\alpha)$ defined in (26),

$$\begin{aligned}
\phi'(0) &= d_n^T(\nabla_x f - X^{-1}\mu - A^T(\lambda - Rc)), \\
\phi''(0) &= d_n^T(H(x, \lambda - Rc) + MX^{-2})d_n + 2d_x^T(\nabla_x f - X^{-1}\mu - A^T(\lambda - Rc)) - 2d_{\lambda\rho}^T c.
\end{aligned}$$

If negative curvature is available in the current iteration, that is, if $d_n \neq 0$, from (24) and $\nabla c(x) d_n = 0$ it follows that $\phi'(0) \leq 0$. We still need to have $\phi''(0) < 0$. If the current values of d_x, d_n, $d_{\lambda\rho}$ and ρ are such that this condition is not satisfied, we set $d_n = 0$ for the search.

If no negative curvature has been detected or the preceding condition has resulted in having $d_n = 0$, then $\phi'(0) = 0$ and conditions (28) and (29) become equivalent to

$$\begin{aligned}
\phi(\alpha) &\leq \phi(0) + \gamma\alpha^2(d_x^T(\nabla f - X^{-1}\mu - A^T(\lambda - Rc)) - d_{\lambda\rho}^T c) \\
\phi'(\alpha) &\geq \eta\alpha(d_x^T(\nabla f - X^{-1}\mu - A^T(\lambda - Rc)) - d_{\lambda\rho}^T c).
\end{aligned}$$

For the curvilinear search to be well defined we need to have (see [15]):

$$d_x^T(\nabla_x f - X^{-1}\mu - A^T(\lambda - Rc)) - d_{\lambda\rho}^T c \leq 0. \tag{35}$$

This condition is a slightly stronger version of the classical descent requirement $d_x^T \nabla_x LA \leq 0$, but (35) must take into account that the multiplier update $\lambda(\alpha)$ (27) is also included in the curvilinear search.

Condition (35) can always be satisfied for an adequate choice of the penalty parameter vector ρ. If at the current iterate the equality constraints are satisfied, $c(x) = 0$, (35) can be rewritten as

$$d_x^T(\nabla_x f - X^{-1}\mu) \leq 0,$$

but from (17) and the positive definiteness of $\bar{G}_\rho$ in the appropriate subspaces,

$$-d_x^T \bar{G}_\rho d_x = d_x^T(\nabla_x f + X^{-1}\mu) \leq 0,$$

and (35) is satisfied.

If $c(x) \neq 0$, (35) may not hold for the current value of the penalty parameter ρ, and it must be modified. We can rewrite (35) as

$$\theta + d_x^T A^T Rc \leq 0, \tag{36}$$

for θ defined as

$$\theta = d_x^T(\nabla_x f - X^{-1}\mu - A^T\lambda) - d_\lambda^T c.$$

From (17), (36) is equivalent to

$$\theta - c^T Rc \leq 0.$$

Define $z \in \mathbb{R}^m$ as $z_j = c_j^2$, and note that $c^T Rc = z^T \rho$. Following the procedure used in code NPSOL [9], the update of the penalty parameter vector ρ is obtained from the following problem:

$$\begin{array}{ll} \min & \frac{1}{2}\rho^T\rho \\ \text{s.t.} & z^T\rho \geq \theta \\ & \rho \geq 0. \end{array} \tag{37}$$

The solution of this problem is given by $\rho^* = \omega z$, where ω is defined as $w = \theta/(z^T z)$.

The j-th component of ρ will be updated to $\delta_\rho \rho_j^*$ (for some $\delta_\rho > 1$), if (35) does not hold and $\rho_j < \delta_\rho \rho_j^*$. In practice, it will also be necessary to ensure that ρ does not become too large (see [10]) to avoid ill-conditioning. If ρ_j is much larger than ρ_j^*, we will reduce its value while ensuring that (35) is still satisfied. The strategy we will follow to update ρ is similar to the one described in [4]. We will compute a trial value $\hat{\rho}_j$:

$$\hat{\rho}_j = \sqrt{\rho_j(\bar{\delta}_\rho + \rho_j^*)},$$

where $\bar{\delta}_\rho \geq 1$, and the new value of ρ at iteration k will be defined as

$$\rho_j^{k+1} = \begin{cases} \delta_\rho \rho_j^* & \text{if } \delta_\rho \rho_j^* > \rho_j^k, \\ \hat{\rho}_j & \text{if } \hat{\rho}_j \leq \frac{1}{2}\rho_j^k, \\ \rho_j^k & \text{otherwise.} \end{cases} \tag{38}$$

To avoid having to modify ρ too often, the parameter $\bar{\delta}_\rho$ is increased at each iteration where ρ is modified.

4.3 The barrier parameters

The vector of barrier parameters in (2) is also updated in each iteration. The updating rule is based on the relationship between the complementarity conditions and the values of the barrier parameters. Let $F(x, \lambda, \sigma)$ be a measure of the satisfaction of the first-order KKT conditions for problem (1) at the current iterate, that is,

$$F(x, \lambda, \sigma) = \begin{pmatrix} \nabla_x f(x) - \nabla c(x)^T \lambda - \sigma \\ c(x) \\ \Sigma x \end{pmatrix}, \tag{39}$$

set

$$\theta = \begin{cases} \|F(x, \lambda, \sigma)\|_2 & \text{if } \|F(x, \lambda, \sigma)\|_2 \geq 1, \\ \|F(x, \lambda, \sigma)\|_2^2 & \text{otherwise,} \end{cases} \tag{40}$$

and define $y = X\sigma$. Vector μ is updated in a similar manner to the penalty parameter ρ. The problem

$$\begin{array}{ll} \min & \frac{1}{2}\mu^T \mu \\ \text{s.t.} & y^T \mu = \theta \\ & \mu \geq 0 \end{array} \tag{41}$$

is solved to obtain $\mu^* = \omega y$, where ω is a scalar defined as $\omega = \theta/(y^T y)$. Definition (40) has been introduced to prevent μ_i^* from becoming too large when far from a KKT point. On the other hand, if y_i is small then μ_i^* may become too small. To avoid this situation we compute a reference value $\hat{\mu}$, similar to the one used in [3],

$$\hat{\mu} = \frac{x^T \sigma}{n},$$

and define the new value of μ at iteration k as

$$\mu_i^{k+1} = \min(\delta^k \max(\mu_i^*, \hat{\mu}), \mu_i^k), \tag{42}$$

where $\delta^k = \min(0.25, \exp(-(1/\theta^k)))$. Note that μ_i will not be decreased in every iteration, but only when a sufficient reduction in the satisfaction of the KKT conditions has been achieved. This definition of μ ensures that $\mu \to 0$ if problem (2) has a solution. The preceding definition of μ_i can be shown to be $O(\|F(x, \lambda, \sigma)\|_2^2)$ near a KKT point, that is one of the conditions required to attain quadratic convergence (see [3] or [23]).

5 Implementation issues

The algorithm described in the preceding sections includes certain parameters and conditions that have not been completely specified yet. In the following paragraphs we indicate how to carry out some of these computations.

5.1 Use of negative curvature directions

After the factorization process has detected the presence of negative curvature in a given iteration, several additional conditions are checked before using this negative curvature direction in the curvilinear search. These conditions are:

$$d_n^T \nabla_{xx}^2 L A d_n \leq -\varepsilon_1, \tag{43}$$

$$d_n^T \nabla_{xx}^2 L A d_n \leq d_n^T G_\rho d_n + \varepsilon_2, \tag{44}$$

$$\|c(x)\| \leq \varepsilon_3, \tag{45}$$

where ε_1, ε_2 and ε_3 are positive constants. Condition (43) guarantees that d_n is a direction of negative curvature for the augmented Lagrangian merit function. Condition (44) discards those cases when (6) is far from being satisfied. Finally, (45) guarantees that we only use negative curvature when we are close enough to feasibility. If any of these conditions is not satisfied, the algorithm sets $d_n = 0$. In our implementation we have defined $\varepsilon_1 = 10^{-7}$ and $\varepsilon_2 = 10^{-3}$. The parameter ε_3 is defined in each iteration as:

$$\varepsilon_3^k = \min(10^{-2}(0.1 + \|x^k\| + |f|), 3),$$

where f is the objective function of (1).

5.2 Convergence criterion

The stopping criterion for the algorithm will be related to the satisfaction of the first and second-order KKT conditions for problem (1). The algorithm will stop if no negative curvature has been detected at the current iteration and the condition

$$\|F(x, \lambda, \sigma)\|_2 \leq \varepsilon(1 + \|\nabla_x f(x)\|)$$

is satisfied at the current iterate. In this condition $F(x, \lambda, \sigma)$ denotes the measure of optimality defined in (39) and $f(x)$ is the objective function for problem (1). We have taken $\varepsilon = 10^{-8}$.

5.3 Initial values of parameters and variables

Let x^0 denote the starting point for the algorithm, assumed to be specified by the user. The only restriction on the values of x^0 will be the satisfaction of the simple bounds. The remaining initial values for the variables and parameters will be defined from x^0. The initial value of the dual variables σ^0 will be defined as $\sigma^0 = (X^0)^{-1}e$. The initial Lagrange multiplier estimate λ^0 will be chosen as an approximation to the least-squares solution of the linear system

$$(A^0)^T\lambda^0 = \nabla_x f(x^0) - \sigma^0.$$

The penalty parameter vector ρ^0 is initially taken to be zero. The initial barrier vector μ^0 is defined using (42) evaluated at the preceding values.

5.4 Other parameters

The constant β_c in (33) is updated recursively. In iteration $k + 1$ it is defined to be

$$\beta_c^{k+1} = \max\left(\sqrt{K\|c(x^k)\|\beta_c^k}, 1\right),$$

where $K > 1$ is a constant (in our implementation we use $K = 7.5$). The initial value of β_c is defined as

$$\beta_c^0 = \max\left(\sqrt{K\|c(x^0)\|}, 1\right).$$

5.5 Numerical difficulties

If a variable gets very close to its corresponding bound at a given iteration, it is possible that due to roundoff errors its value may be considered to be equal to the bound and the logarithmic function will not be defined in subsequent iterations. A possible solution is given in [7], where the variables are forced to remain apart from their bounds by a fixed distance, chosen as 10^{-9}. However, this strategy presents a clear disadvantage: the solution of a particular problem, for a sufficiently small value of μ, might be closer to the bound than the previous tolerance. In our numerical experience, this option may delay convergence by a significant number of iterations.

In our strategy, the information on the distance to the bounds will be kept in a vector r, that will be updated separately from the values of the variables, using the same information. The independent term in inequality (30) will also be defined in terms of this vector.

Another numerical problem that might arise is the ill-conditioning of the symmetric system (17) to solve in each iteration, due to the terms $X^{-1}\Sigma$. Under reasonable conditions, it can be shown that this ill-conditioning is benign (see [19] and [21]).

5.6 The algorithm

We present a scheme of the proposed interior point algorithm (**C**urvilinear **S**earch **I**nterior **P**oint **M**ethod - **CSIPM**), summarizing the aspects described in the previous sections.

Algorithm CSIPM

Choose initial values for x^0, λ^0 *and* σ^0.
Choose initial values for vectors ρ^0 *and* μ^0
Set $k = 0$
repeat
  Compute d_x^k, d_λ^k *and* d_σ^k *from (17) using the factorization described in [6]*
  Compute $d_{\lambda\rho}^k$ *from (25)*
  Compute, if it exists, d_n^k, *a direction of negative curvature from (23)*
  Compute ρ^{k+1} *from (38)*
  Set $d_n^k = 0$ *if any one of the conditions (43), (44) or (45) is not satisfied*
  Compute α^p *using the curvilinear search procedure satisfying (28), (29) and (33)*
  Compute α^d *from (34)*
  $x^{k+1} = x^k + (\alpha^p)^2 d_x^k + \alpha^p d_n^k$
  $\lambda^{k+1} = \lambda^k + (\alpha^p)^2 d_{\lambda\rho}^k$
  $\sigma^{k+1} = \sigma^k + \alpha^d d_\sigma^k$
  Compute the updated barrier vector μ^{k+1} *from (42)*
  $k = k + 1$
until *convergence*

6 Numerical results

We have conducted a set numerical experiments on a collection of test problems using algorithm CSIPM. The algorithm has been implemented, and the tests have been carried out, on MATLAB 4.3.

In the preceding description of the algorithm we have considered only simple bounds of the form $x \geq 0$, to simplify the resulting expressions. The implementation of the algorithm used for the tests is able to handle simple bounds of the form $l \leq x \leq u$, where some of the entries in l could be equal to $-\infty$, and some of those in u could be ∞. These bounds are included in the objective function via logarithmic barrier terms.

6.1 Small sized problems (less than 100 variables)

The first subset we have considered is composed of 36 small problems from the Hock and Schittkowski collection, [11]. We have selected all problems with more than 6 variables and a sparse Hessian of the Lagrangian function (note that exact first and second derivatives have been used). Moreover, some strongly nonconvex problems have also been selected (HS64 and HS72, see [8]), as well as problem HS13 whose Jacobian is rank deficient at the solution. Whenever possible, the initial points proposed in [11] have been taken. Sometimes these initial points do not satisfy the bound constraints. Such points have been transformed following a strategy similar to that described in [18]. Table 1 shows the results obtained by CSIPM for these small problems. The columns in the table correspond to:

- `Prob.`: problem name.

- `Const.`: norm for the constraint vector including slacks.
- `KKT`: norm for the first-order KKT conditions including slacks.
- `Iter.`: iteration count (number of factorizations of the primal-dual system).
- `Eval.`: number of evaluations of the objective function and the constraints..
- `NC`: number of iterations where directions of negative curvature were used.

Table 1: Results for small-size problems

Prob.	Obj.	Const.	KKT	Iter.	Eval.	NC
HS13	--	--	--	--	--	--
HS32	1.00000001	7.9e-13	7.0e-08	14	14	0
HS53	4.0930232	1.8e-15	6.2e-14	4	4	0
HS64	6299.84243	1.2e-16	2.5e-14	26	30	0
HS65	0.95352886	9.8e-32	1.4e-15	15	26	2
	0.95352886	2.1e-08	2.8e-11	17	31	0
HS71	17.0140173	2.2e-14	3.7e-14	8	8	0
HS72	727.67936	1.7e-18	7.3e-12	24	24	0
HS73	29.894378	8.0e-15	1.2e-13	16	16	0
HS74	5126.4981	1.4e-11	6.5e-09	8	8	0
HS75	5174.4127	2.5e-13	4.3e-10	9	9	0
HS80	0.0539498	9.3e-09	9.9e-09	9	9	0
HS81	0.0539498	1.5e-10	1.6e-10	9	9	0
HS83	-30665.539	3.4e-09	1.2e-12	19	19	0
HS84	-5280335.13	1.0e-08	1.8e-07	36	43	0
HS86	-32.348679	1.0e-14	1.0e-08	14	14	0
HS93	135.075963	3.2e-15	1.5e-07	9	9	0
HS95	0.0156195	6.9e-12	2.7e-10	11	11	0
HS96	0.0156195	2.6e-11	8.3e-10	11	11	0
HS97	4.0712463	6.2e-12	2.0e-09	13	13	0
HS98	4.0712463	9.7e-15	1.2e-12	16	19	1
	4.0712463	1.0e-10	2.0e-08	16	17	0
HS99	-8.3108e8	4.4e-11	0.49945913	6	6	0
HS100	680.630057	9.0e-10	2.3e-09	9	13	1
	680.630057	1.9e-09	5.1e-09	10	15	0
HS104	3.9511634	1.5e-13	1.0e-12	12	12	0
HS106	7049.24802	4.1e-11	4.1e-11	13	14	0
HS107	4797.98185	5.2e-14	7.5e-09	9	13	0
HS108	-0.8660254	3.2e-10	5.3e-10	19	24	0
HS109	--	--	--	--	--	--
HS110	-45.7784697	--	4.8e-13	5	5	0
HS111	-47.7610917	5.0e-08	9.0e-08	12	18	0
HS112	-47.7610908	2.5e-06	1.8e-08	12	18	0
HS113	24.306209	1.4e-11	2.3e-11	18	25	3
	24.306209	5.1e-09	5.8e-08	41	54	0
HS114	-1768.80696	3.1e-11	1.0e-10	17	18	0
HS116	97.5875096	2.9e-13	8.4e-11	32	36	0
HS117	32.3486790	1.2e-10	5.7e-10	23	27	0
HS118	664.820450	2.0e-14	3.1e-10	14	14	0
HS119	244.899697	2.2e-16	2.9e-08	12	12	0

In those cases were negative curvature was detected (HS65, HS98, HS100 and HS113) the problem was solved a second time, without using negative curvature. An improvement was noticed for most of the

problems when negative curvature was used; this improvement was particularly remarkable for problem HS113. The algorithm is able to solve 34 problems, but it fails to solve two of them, problems HS13 (with a rank-deficient Jacobian at the solution) and HS109 (after exceeding 250 iterations no solution was reached). For four problems the code finds better local minimizers than those given in [11] (problems HS106, HS107, HS112 and HS116), while for two of the problems the local minimizers found are worse (HS97 and HS98). Problem HS99 is an example of a badly scaled problem. The termination tolerance is satisfied when the norm of the first-order KKT conditions is 0.4994. Introducing a more demanding stopping criterion (a tolerance of 10^{-14}), the norm of the KKT conditions goes down to 10^{-6} after 3 additional iterations, but the value of the merit function remains basically unaltered.

6.2 Medium sized problems (100 to 400 variables)

We have also tested the algorithm on two medium sized problems, instances of multi-area AC Optimal Power Flow (OPF) problems. These problems are OPF24 and OPF24-3; Table 2 provides some information on their properties and a more detailed description can be found in [12] and [22], respectively. The columns in this table correspond to:

- `Prob.`: problem name.
- `Var.`: number of variables.
- `EC`: number of equality constraints.
- `IC`: number of inequality constraints (bounds not included).
- `LB`: number of lower bound constraints.
- `UC`: number of upper bound constraints.
- `TC`: total number of constraints (bounds included).
- `Sis.`: size of the primal-dual system (17).

Table 2: Description of medium-size problems

Prob.	Var.	EC	IC	LB	UB	TC	Sis.
OPF24	112	48	34	112	112	306	228
OPF24-3	336	144	107	336	336	923	694

Table 3 shows the results obtained for these problems. The algorithm is able to solve both problems in a reduced number of iterations. Note that the number of iterations required by the procedure using negative curvature is larger than that required when no negative curvature is used, for problem OPF24-3. The reason is that the minimizers computed in both cases are different. This is a consequence of the nonconvexity of the problems, and the existence of several local minimizers.

Table 3: Results for medium-size problems

Prob.	Obj.	Const.	KKT	Iter.	Eval.	NC
OPF24	4344.8315	6.3e-13	1.0e-07	28	32	0
OPF24-3	110.53257	1.9e-10	2.8e-08	31	32	9
	110.46529	9.0e-11	2.1e-10	26	26	0

7 Conclusions

In this paper we have described an efficient procedure to compute local solutions of nonconvex problems. The procedure is based on a primal-dual method to define the search directions, and a curvilinear search to combine them.

The implemented version of the algorithm has been tested on a set of small and medium test problems. The results show that the procedure works quite well on these problems, compared to other proposals in the literature. The use of vector penalty and barrier parameters is in part responsible for this good behavior.

The impact of the negative curvature is not very significant on these small problems, but it can be quite significant in those few cases when it is used. As would be expected, there is an increasing tendency to use negative curvature information as the problems get larger. Given the limited cost of computing a direction of negative curvature whenever an appropriate factorization is used to obtain the movement directions, it would seem that any reasonable algorithm should allow the use of this second-order information.

Acknowledgements

We wish to thank A. Forsgren and M. H. Wright for their helpful comments and suggestions.

References

[1] Bertsekas, D. P. *Constrained optimization and Lagrange multiplier methods.* Academic Press, New York, 1982.

[2] Bunch, J. R., Kaufman, L. and Parlett, B. N. "Decomposition of a symmetric matrix", *Numer. Math.* **27**, pp. 95–109, 1976.

[3] El-Bakry, A. S., Tapia, R. A., Tsuchiya, T. and Zhang, Y. "On the formulation and theory of the Newton interior-point method for nonlinear programming", *Journal Optim. Theory Applications* **89**, pp. 507–541, 1996.

[4] Eldersveld, S. K. *Large-scale sequential quadratic programming algorithms.* Technical Report SOL 92-4. Stanford University, 1992.

[5] Fiacco, A. V. and McCormick, G. P. *Nonlinear programming: Sequential unconstrained minimization techniques.* Society for Industrial And Applied Mathematics, Philadelphia, 1990 (Originally published by Research Analysis Corporation, McLean, Virginia).

[6] Forsgren, A. and Murray, W. "Newton methods for large-scale linear equality-constrained minimization." *SIAM J. on Matrix Analysis and Applications* **14**, pp. 560–587, 1993.

[7] Gajulapalli, R. S. *INTOPT: An interior point algorithm for large scale nonlinear optimization.* Ph. D. Thesis. The University of Texas at Austin, 1995.

[8] Gay, D. M., Overton, M. L. and Wright, M. H. *A primal-dual interior method for nonconvex nonlinear programming.* Technical Report 97-4-08. Computing Science Research Center, Bell Laboratories, Murray Hill, New Jersey, 1997.

[9] Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. *User's guide for NPSOL (version 4.0): A FORTRAN package for nonlinear programming.* Technical Report SOL 86-2. Stanford University, 1986.

[10] Gill, P. E., Murray, W. and Wright, M. H. *Practical optimization.* Academic Press, London/New York, 1981.

[11] Hock, W. and Schittkowski, K. *Test examples for nonlinear programming codes.* Springer Verlag. Berlin, 1981.

[12] IEEE Committee Report. "IEEE reliability test system", *IEEE Transactions on Power Systems* **98**, pp. 2047–2054, 1979.

[13] McCormick, G. "A modification of Armijo's step-size rule for negative curvature", *Math. Programming* **13**, pp. 111–115, 1977.

[14] Mehrotra, S. "On the implementation of a primal-dual interior point method", *SIAM J. Optim.* **2**, pp. 575–601, 1992.

[15] Moré, J. J. and Sorensen, D. C. "On the use of directions of negative curvature in a modified Newton method", *Math. Programming* **16**, pp. 1–20, 1979.

[16] Murray, W. and Prieto, F. J. *A second-derivative method for nonlinearly constrained optimization.* Technical Report SOL 95-3. Stanford University, 1995.

[17] Stewart, G. W. and Sun, J. *Matrix perturbation theory.* Academic Press, Boston, 1990.

[18] Vanderbei, R. J. and Shanno, D. F. *An interior-point algorithm for nonconvex nonlinear programming.* Technical Report SOR-97-21, Princeton University, 1997.

[19] Wright, M. H. *Ill-conditioning and computational error in primal-dual methods for nonlinear programming.* Technical Report 97-4-04. Computing Science Research Center, Bell Laboratories, Murray Hill, New Jersey, 1997.

[20] Wright, M. H. "Some properties of the Hessian of the logarithmic barrier fuction", *Math. Programming* **67**, pp. 265–295, 1994.

[21] Wright, S. J. *Effects of finite-precision arithmetic on interior-point methods for nonlinear programming.* Preprint ANL/MCS-P705-0198. Mathematics and Computer Science Division, Argonne National Laboratory, 1998.

[22] Xia, F. and Sakis-Meliopoulos, A. P. "A methodology for probabilistic simultaneous transfer capability analysis", *IEEE Transactions on Power Systems* **11**, pp. 1269–1276, 1996.

[23] Yamashita, H and Yabe, H. "Superlinear and quadratic convergence of some primal-dual interior point methods for constrained optimization", *Math. Programming* **75**, pp. 377–397, 1996.