Javier M. Moguerza · Francisco J. Prieto

Combining search directions using gradient flows

Abstract. The efficient combination of directions is a significant problem in line search methods that either use negative curvature, or wish to include additional information such as the gradient or different approximations to the Newton direction.

In this paper we describe a new procedure to combine several of these directions within an interior-point primal-dual algorithm. Basically, we combine in an efficient manner a modified Newton direction with the gradient of a merit function and a direction of negative curvature, if it exists. We also show that the procedure is well-defined, and it has reasonable theoretical properties regarding the rate of convergence of the method.

We also present numerical results from an implementation of the proposed algorithm on a set of small test problems from the CUTE collection.

Key words. negative curvature – primal-dual methods – interior-point methods – nonconvex optimization – line searches

1. Introduction

Our goal is to describe a procedure to combine search directions in algorithms that compute local solutions for nonlinear optimization problems of the form

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

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$, and we assume all functions to be at least twice continuously differentiable.

Standard procedures for computing these solutions are based on solving local approximations to the problem at successive iterates x_k, and using the solutions for these approximations as the next iterates. This basic scheme is complicated by the need to ensure global convergence for the resulting algorithm, attained basically through the use of trust-region schemes or searches along a parametrized line or curve (linesearch methods); we will concentrate on procedures that are based on this second alternative. Standard algorithms in this class (see for example Fletcher [14], Gill et al. [18]) rely

J. M. Moguerza: School of Engineering, Univ. Rey Juan Carlos, Madrid, Spain,
e-mail: `j.moguerza@escet.urjc.es`.
Research supported by Spanish MEC grant TIC2000-1750-C06-04 and CAM project 07T/0005/2001

F. J. Prieto: Dept. of Statistics and Econometrics, Univ. Carlos III de Madrid, Spain,
e-mail: `fjp@estecon.uc3m.es`.
Research supported by Spanish MEC grants BEC2000-0167 and PB98-0728

Mathematics Subject Classification (1991) 49M37, 65K05, 90C30

on some modified version of the Newton direction and compute the next iterate on the unidimensional subspace generated by this direction. In some cases, it may be useful or efficient to take into account additional information to determine this next iterate. For example, the use of directions of negative curvature is needed to ensure the convergence of the algorithm to second-order KKT points, while perhaps improving the efficiency of the algorithm. The use of additional descent information, obtained from the gradient, for example, may provide more robust algorithms and may yield better iterates, particularly away from the solution. Our goal is to derive a procedure that, by taking into account additional search information in an efficient manner, requires a reduced number of iterations to obtain a solution for problem (1), with a consequently reduced computational cost.

Although these directions might be used in the line searches independently of each other to generate the sequence of iterates, it seems more efficient to combine them before computing the next iterate. Many proposals based on these ideas can be found in the literature. The dogleg method (see Dennis and Schnabel [10], for example) combines the gradient and Newton direction in an attempt to mimic the behavior of trust-region methods. Moré and Sorensen [23] proposed a procedure to find the next iterate from a direction of descent and one of negative curvature by following a quadratic curve on the subspace spanned by both directions.

In this paper we present a proposal based on the approximate solution of a system of ordinary differential equations. This idea was first proposed in Courant [8]; other related proposals can be found in Behrman [1], Botsaris [4], Del Gatto [9], Schropp [24] and Zang [29] for the unconstrained case, and Evtushenko and Zhadan [12] for the constrained case. Our proposal is most closely related to that in [1] for the unconstrained case, where the iterates were found by constructing a Krylov subspace in each iteration, performing a standard univariate search on the steepest descent curve defined on this subspace. We apply similar ideas to the combination of the search directions in a constrained optimization setting. The paper does not contain a global convergence analysis for this proposal, although we have included some comments on the rate of convergence of the algorithm.

The main difficulty when combining the different search directions arises from the differences in the scales of these directions. While the Newton direction is in general well scaled (a step of one is reasonable in many cases, at least when close to the solution), this is not true either for directions of negative curvature or for the gradient direction, our choices of additional search directions. One alternative to overcome this difficulty would be to carry out a search on the reduced-dimension subspace spanned by these search directions. Byrd et al. [6] compute the next iterate from a linear combination of a direction of negative curvature and a gradient direction, and these coefficients are obtained as the solution of a two-dimensional trust-region problem. Nevertheless, most proposals in the literature reduce first the search to a univariate one, to attain greater computational efficiency. This will also be our approach; we will construct a curve in the subspace generated by the directions of interest: descent direction, negative curvature and gradient. A reasonable curve in this subspace would be the one that corresponds to the trajectory having the steepest descent at each point; this trajectory would lead to a local solution at the fastest rate, measured in terms of the objective function. Unfortunately, this curve is in general very expensive to compute, and we will satisfy ourselves with constructing a steepest descent curve based on a simple (quadratic) local model of the problem. The next iterate is obtained as a point on the curve providing sufficient descent for an appropriate merit function.

This proposal will be introduced as part of a complete algorithm for the solution of problem (1), based on a primal-dual interior point method and the use of an augmented Lagrangian merit function. The method computes approximate solutions for a sequence of barrier problems of the form

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

where $\mu_k \to 0$. Note that we use a vector of barrier parameters $\mu \in \mathbb{R}^n_+$. See Fiacco and McCormick [13], Wright [26] for a theoretical analysis of a similar procedure based on a scalar barrier parameter, $\mu \in \mathbb{R}_+$.

In each iteration, the algorithm computes a descent direction and a direction of negative curvature for problem (2), if it exists. The search directions are obtained from the application of Newton's method to the primal-dual equations for problem (2), see [11] for example,

$$\begin{aligned} \nabla f(x) - \nabla c^T(x)\lambda - z &= 0, \\ c(x) &= 0, \\ Zx &= \mu, \end{aligned} \tag{3}$$

where the nonnegative dual variables z correspond in the limit to the multipliers for the simple bounds in (1), and $Z = \operatorname{diag}(z)$.

These directions and the gradient of an augmented Lagrangian merit function (see Bertsekas [2])

$$L_A(x, \lambda; \mu, \rho) = f(x) - \sum_i \mu_i \log x_i - \lambda^T c(x) + \frac{\rho}{2}\|c(x)\|^2, \tag{4}$$

are then combined to generate a new iterate that provides sufficient decrease for this merit function. This merit function has been extensively used in optimization packages, see for example Conn et al. [7]. The derivatives of the merit function will play a significant role in the definition of the algorithm; we indicate their expressions for future reference:

$$\nabla_x L_A = \nabla f(x) - X^{-1}\mu - \nabla c(x)^T\lambda + \rho\nabla c(x)^T c(x) \tag{5}$$

$$\nabla_{xx} L_A = \nabla^2 f(x) - \sum_j (\lambda_j - \rho c_j(x))\nabla^2 c_j(x) + MX^{-2} + \rho\nabla c(x)^T\nabla c(x), \tag{6}$$

where $X = \operatorname{diag}(x)$ and $M = \operatorname{diag}(\mu)$.

The rest of the paper is organized as follows: In Section 2 we describe the general algorithm to compute a local solution for problem (1). In Section 3 we motivate and describe the proposal to combine the directions generated by the algorithm to obtain the next iterate. In Section 4 we justify some basic properties of this procedure, such as for example that it is well-defined, that sufficient descent can be achieved in each iteration and that the algorithm attains superlinear convergence. Finally, Section 5 gives the general structure of the algorithm, discusses some implementation issues and presents and comments some computational results on a set of small test problems.

2. The interior-point algorithm

Our main goal is to explore an alternative procedure for the combination of search directions in a line-search based algorithm. In this regard, we are interested in determining the impact this approach may have in the practical behavior of a nonlinear optimization algorithm. As a consequence, we will introduce an algorithm that uses the combination procedure of interest, but that also computes efficiently the required search directions. This algorithm is based on a primal-dual interior point approach to generate the search directions, and uses a line search procedure on an augmented Lagrangian merit function to ensure global convergence. An iterative algorithm of this sort carries out three main tasks in each iteration: i) Compute search directions at the current iterate. In our case we will obtain a descent direction and a direction of negative curvature from the KKT system of linear equations, in addition to the gradient of the merit function. ii) Combine the directions to obtain the next iterate. iii) Update the parameters in the algorithm.

```
Initialize variables ($x^0, \lambda^0, z^0$), barrier ($\mu^0$) and
      penalty ($\rho^0$) parameters
repeat
      From the Newton primal-dual equations:              Section 2.1
            Compute a descent direction, $(d_x)_k$,
                  for the primal variables $x$
            Compute directions, $(d_\lambda)_k$ and $(d_z)_k$,
                  for the multipliers $\lambda$ and $z$
      Compute $d_g$ as the gradient of the
            merit function                                Section 2.1
      Compute, if it exists, $(d_n)_k$, a direction
            of negative curvature                         Section 2.1
      Adjust the penalty parameter $\rho_k$                Section 3.4
      Combine the three directions $d_x$, $d_n$ and $d_g$      Section 3.1
      Update the primal variables                         Section 3.2
      Update the multipliers                              Sections 3.3, 2.2.1
      Decrease the barrier parameter vector $\mu_k$        Section 2.2.2
until convergence                                         Section 5.2
```

The preceding table presents a schematic version of the algorithm, indicating the Sections of the paper where the different steps are described. Section 3 will be devoted to our main concern, the description of the combination of directions, while in this section we will describe those issues related to the first and third items, providing only the basic details of the procedures implemented in the algorithm. Additional information can be found in Moguerza and Prieto [22].

2.1. Computing the search directions

In the proposed algorithm we solve a sequence of problems (2) such that $\mu_i \to 0$ for all i, following [13]. The search directions are obtained from the application of Newton's method to the primal-dual equations (3). It provides directions d_x, d_λ and d_z, corresponding to updates for the variables x, λ and z respectively. From the first-order Taylor series expansion for the primal-dual KKT conditions (3) about the current values x, λ and z, the resulting system of linear equations defining the search directions is (we omit the dependence on the variables to simplify the notation):

$$\begin{pmatrix} H & -\nabla c^T & -I \\ \nabla c & 0 & 0 \\ Z & 0 & X \end{pmatrix} \begin{pmatrix} d_x \\ d_\lambda \\ d_z \end{pmatrix} = \begin{pmatrix} -\nabla f + \nabla c^T \lambda + z \\ -c \\ \mu - Zx \end{pmatrix}, \tag{7}$$

where $H = \nabla_{xx} L(x, \lambda)$, $L(x, \lambda)$ is the Lagrangian function for problem (1), that is, $L(x, \lambda) = f(x) - \lambda^T c(x) - \omega^T x$, and I denotes the identity matrix. From the last set of equations in (7), we have

$$d_z = X^{-1}\mu - z - X^{-1} Z d_x. \tag{8}$$

Replacing (8) into the first two sets of equations in (7), the movement direction d_x can be computed as the solution of the symmetric system

$$K \begin{pmatrix} d_x \\ -d_\lambda \end{pmatrix} = \begin{pmatrix} -\nabla f + \nabla c^T \lambda + X^{-1}\mu \\ -c \end{pmatrix}, \tag{9}$$

where K is defined as

$$K = \begin{pmatrix} G & \nabla c^T \\ \nabla c & 0 \end{pmatrix}, \tag{10}$$

for $G = H + X^{-1} Z$. Any ill-conditioning that might arise from the diagonal terms in G is benign, see Wright [27] for example.

The direction obtained from (9) 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. We adapt system (9) to ensure that the direction d_x is a sufficient descent direction for the merit function (4). The modified system that we use to define these search directions is

$$\begin{pmatrix} \bar{G}_\rho & \nabla c^T \\ \nabla c & 0 \end{pmatrix} \begin{pmatrix} d_x \\ -d_\lambda \end{pmatrix} = \begin{pmatrix} -\nabla f + \nabla c^T \lambda + X^{-1}\mu \\ -c \end{pmatrix}, \tag{11}$$

where its coefficient matrix (and $\bar{G}_\rho$ in particular) is computed from a modification of

$$K_\rho = \begin{pmatrix} G_\rho & \nabla c^T \\ \nabla c & 0 \end{pmatrix}, \tag{12}$$

for $G_\rho = \nabla_{xx} L(x, \lambda - \rho c) + X^{-1} Z$. The motivation for this definition of G_ρ, instead of using G as in (9), will be discussed in terms of the procedure to combine the search directions, in Section 3.1.

To justify replacing G_ρ with $\bar{G}_\rho$, consider the directional derivative for the merit function along the search directions d_x and d_λ. From (5) and (11) we have

$$\begin{aligned} \nabla_x L_A^T d_x &= (\nabla f(x) - X^{-1}\mu - \nabla c(x)^T (\lambda - \rho c(x)))^T d_x \\ &= -d_x^T \bar{G}_\rho d_x - c(x)^T d_\lambda - \rho \|c(x)\|^2. \end{aligned}$$

This directional derivative can be made sufficiently negative by increasing the value of ρ whenever $c(x) \neq 0$ (and d_λ is bounded). Otherwise, it is enough that $\bar{G}_\rho$ is sufficiently positive definite in the nullspace of $\nabla c(x)$, as in this case $\nabla c(x) d_x = -c(x) = 0$.

An appropriate matrix $\bar{G}_\rho$ for (11), such that $W_A^T \bar{G}_\rho W_A$ is positive definite, can be generated in the process of factorizing K_ρ, where W_A has columns that form a basis for the null-space of $\nabla c(x)$. A modified Cholesky factorization of the reduced Hessian $W_A^T G_\rho W_A$ could be used, as in Gay et al. [16]. This approach requires forming explicitly the reduced Hessian, and as a consequence it is only useful for problems in which this reduced Hessian is not too large. We have chosen to use a version of the symmetric indefinite factorization, see Bunch et al. [5] for example, incorporating the modifications proposed in Forsgren and Murray [15]. This alternative is able to obtain the desired modification for the reduced Hessian directly from system (11), it allows the computation of appropriate directions of negative curvature, as we will indicate below, and it can be applied to medium-sized and large problems. Additional details of the computation of these directions and the factorization used in the algorithm can be found in [15, 22].

We also want to satisfy the necessary second-order condition at any limit point. For problem (2) this condition requires that

$$W_A^T \left(\nabla_{xx} L(x, \lambda) + M X^{-2} \right) W_A \quad \text{is p.s.d.} \tag{13}$$

We will use directions of negative curvature to avoid converging to points that do not satisfy (13), but as the direction of negative curvature is used to obtain iterates that decrease the merit function (4), we also need to ensure that such a direction is appropriate for our merit function. A direction of negative curvature d_n for our algorithm should lie in the subspace spanned by the columns of W_A and should satisfy

$$d_n^T (\nabla_{xx} L(x, \lambda - \rho c) + M X^{-2}) d_n < 0. \tag{14}$$

Note that from (6) if $\nabla c(x) d_n = 0$ then

$$d_n^T \nabla_{xx} L_A(x, \lambda; \mu, \rho) d_n = d_n^T (\nabla_{xx} L(x, \lambda - \rho c) + M X^{-2}) d_n.$$

As we will justify in Section 3, in our case the choice of an appropriate sign for d_n (to ensure descent, for example) is not relevant, as the search for the next iterate is performed on a subspace spanned by a combination of directions including d_n, and the combination chosen by the algorithm will take the best sign into account automatically.

Such a direction d_n (assuming there exists one) is computed from the same symmetric indefinite factorization used to obtain the descent direction d_x from (11). Let K_ρ be the matrix defined in (12), and assume that its symmetric indefinite factorization $K_\rho = U^T D U$ has been computed using the algorithm in [15]. Assume also that from the factorization it has been determined that this matrix has more than m negative eigenvalues, implying that $W_A^T G_\rho W_A$ has at least one negative eigenvalue. Let P be the permutation matrix associated with the pivoting choices in the factorization algorithm and define $w = P\tilde{w}$, where $\tilde{w}$ satisfies

$$\begin{pmatrix} U_{11} & U_{21} \\ 0 & U_{22} \end{pmatrix} \begin{pmatrix} \tilde{w}_1 \\ \tilde{w}_2 \end{pmatrix} = \pm\sqrt{-\lambda_{\min}(D_2)} \begin{pmatrix} 0 \\ u_\lambda \end{pmatrix}, \tag{15}$$

for a partition of U and D such that D_1 and U_{11} correspond to all the pivots taken from elements of ∇c, $\lambda_{\min}(D_2)$ denotes the most negative eigenvalue of D_2 and u_λ is a unit eigenvector corresponding to this smallest eigenvalue. The direction of negative curvature d_n is defined as the first n components of w. Additional details can be found in [15]; in particular, it is shown there that $\nabla c(x) d_n = 0$, and consequently d_n lies in the correct subspace. Also, there exist positive constants k_1 and k_2 such that

$$d_n^T G_\rho d_n \le -k_1 \lambda_{\min}^2 (W_A^T G_\rho W_A) \qquad \text{and} \qquad d_n^T d_n \le -k_2 \lambda_{\min}(W_A^T G_\rho W_A).$$

The direction of negative curvature computed from (15) satisfies

$$d_n^T (G_\rho + X^{-1} Z) d_n < 0.$$

Although this condition is not sufficient to ensure that (14) will be satisfied, the procedure used to obtain the direction of negative curvature allows both d_x and d_n to be computed from the same system of equations in an efficient manner. On the other hand, if $z - X^{-1}\mu$ is sufficiently small then $X^{-1}Z$ is close to MX^{-2}, and if the constraints are close to zero, $\nabla c(x) d_n$ will also be close to zero. Under these conditions the direction d_n computed using the preceding procedure will satisfy $d_n^T \nabla_{xx} L_A d_n < 0$ and (14). In particular, this will happen close to a first-order KKT point for problem (2).

Nevertheless, it is important to ensure the satisfaction of (14) before d_n is used. As in general $z - X^{-1}\mu$ may not be small, each time a direction of negative curvature is computed we will also check if (14) is satisfied. If this is not the case, the direction of negative curvature d_n will not be used.

A third search direction that is used to generate the next iterate is the gradient of the merit function. From (5) this direction is defined as

$$d_g = \nabla_x L_A(x, \bar\lambda; \mu, \bar\rho) = \nabla f - X^{-1}\mu - \nabla c^T (\bar\lambda - \bar\rho c), \tag{16}$$

where $\bar\lambda$ and $\bar\rho$ will be introduced later on.

2.2. *Updating the parameters*

In each iteration the algorithm must update the different parameters involved in the specification of the barrier subproblems (2) and the merit function (4). In the following paragraphs we describe the procedures used to change the multiplier estimates and the barrier parameters.

2.2.1. 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 z. The multipliers λ are updated using d_λ from (11), as described in Section 3.

The solution of the Newton equations (7) provides a search direction for the multipliers z, d_z, defined in (8). These dual variables are updated from

$$z_{k+1} = z_k + (\alpha_d)_k (d_z)_k.$$

The only restriction 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

$$(\alpha_d)_k = \min\left(\tau_k \min\left(\frac{-(z_k)_i}{(d_z)_{ki}}\,\middle|\, (d_z)_{ki} < 0\right), 1\right), \tag{17}$$

where τ_k is defined as

$$\tau_k = \max(0.995, 1 - \|\mu_k\|_2). \tag{18}$$

This definition is introduced to ensure reasonable local convergence properties for the algorithm.

2.2.2. 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 satisfaction of the first-order conditions, the complementarity conditions and the previous values of the barrier parameters. Let $F(x, \lambda, z; \rho)$ be a measure of the satisfaction of the first-order KKT conditions for problem (1) at the current iterate, that is,

$$F(x, \lambda, z; \rho) = \begin{pmatrix} \nabla f(x) - \nabla c(x)^T(\lambda - \rho c) - z \\ c(x) \\ Zx \end{pmatrix}, \tag{19}$$

set

$$\theta = \begin{cases} \|F(x, \lambda, z; \rho)\|_2 & \text{if } \|F(x, \lambda, z; \rho)\|_2 \geq 1, \\ \|F(x, \lambda, z; \rho)\|_2^2 & \text{otherwise,} \end{cases} \tag{20}$$

and define $y = Xz$.

The new value for μ is chosen to ensure a reasonably uniform allocation of the distance from optimality taking into account each complementarity gap. These new values are obtained, in a manner similar to the procedure in [22], from the solution of the problem

$$\begin{array}{ll} \min_\mu & \frac{1}{2}\mu^T\mu \\ \text{s.t.} & y^T\mu = \theta \\ & \mu \geq 0. \end{array} \tag{21}$$

This solution is given by $\mu^* = \omega y$, where $\omega = \theta/(y^T y)$. Definition (20) 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 that in El-Bakry et al. [11],

$$\hat{\mu} = \frac{x^T z}{n}, \tag{22}$$

and define the new value of μ at iteration k as

$$(\mu_{k+1})_i = \min(\delta_k \max(\mu_i^*, \hat{\mu}), (\mu_k)_i), \tag{23}$$

where $\delta_k = \min(0.25, \exp(-(1/\theta_k)))$ and θ_k is given by (20). 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.

3. The computation of a new iterate

We now describe how to combine in an efficient manner our search directions: descent d_x, negative curvature d_n (if it exists) and gradient d_g. Classical line search methods compute a direction of movement (d_x, d_λ) and a scalar α such that the next iterate $(x + \alpha d_x, \lambda + \alpha d_\lambda)$ provides sufficient decrease for an appropriate merit function. This approach works quite well in practice whenever there is a single search direction d_x. In our case we may have up to three search directions at a given iteration, and the preceding procedure must be modified to take into account that we search the next iterate on a subspace of dimension three, as opposed to the univariate classical approach. Following our previous discussion, we proceed first by combining these directions into a trajectory of points of interest, and we then perform a conventional univariate search (a backtracking search) on this trajectory.

3.1. Combining the search directions

In the unconstrained case, and given our three search directions, it would seem reasonable to select the new iterate as a point on the trajectory defined by the steepest descent of the objective function from the current iterate, see [1] for example. For our constrained problem (2), we have chosen to apply these ideas to our merit function (4). In particular, we construct this trajectory from the gradient field of the merit function starting from a given iterate x_k. The trajectory is given by the solution of the system of ordinary differential equations (we omit the iteration subscript to simplify the notation)

$$\dot{\gamma}(t) = -\nabla_x L_A \left(x + \gamma(t), \bar{\lambda}; \mu, \bar{\rho}\right), \quad \gamma(0) = 0, \tag{24}$$

where the reference value for the multipliers, $\bar{\lambda}$, and the penalty parameter value, $\bar{\rho}$, will be defined later on.

Computing this trajectory is too expensive for most practical cases; we will restrict ourselves to solving an approximation to it, in the following two senses:

– We will approximate locally the right-hand side of the first part of (24) by a linear function, to obtain a linear system of ODEs, having a closed-form solution.
– We will also restrict ourselves to those points lying on the subspace spanned by our three search directions, to reduce the dimension of the problem and to limit the computational cost.

From (6) and (16), the local linear approximation to the first part of (24) is given by

$$\begin{aligned}\dot{\eta}(t) &= -\nabla_x L_A - \nabla_{xx} L_A \eta(t) \\ &= -d_g - (\nabla_{xx} L(x, \bar{\lambda} - \bar{\rho} c) + M X^{-2} + \rho \nabla c^T \nabla c)\eta(t).\end{aligned} \tag{25}$$

Note that the Hessian matrix $\nabla_{xx} L(x, \bar{\lambda} - \bar{\rho} c)$ in this approximation coincides with that in the definition of G_ρ, introduced in (11). In Section 4 it will be shown that using $\bar{G}_\rho$ instead of G to compute d_x from (11) allows the final point of the approximate trajectory to be $x + d_x$, the Newton step.

From (25), the ODE that defines the modified trajectory $\beta(t)$ on the two- or three-dimensional subspace of interest will have the following form:

$$\dot{\beta}(t) = -B^T d_g - B^T(V + \bar{\rho}\nabla c^T \nabla c)B\beta(t), \quad \beta(0) = 0, \tag{26}$$

where B denotes an orthonormal basis for the subspace spanned by the search directions and $d_g = \nabla_x L_A(x, \bar{\lambda}; \mu, \bar{\rho})$. The matrix V is equal to G_ρ, as defined in (12), whenever a direction of negative curvature is available, to ensure that this negative curvature is also present along the curve. If no negative curvature is available or it has been discarded, then V is chosen as $\bar{G}_\rho$. In the absence of negative curvature we wish to preserve the property that the final point coincides with the Newton step, and this property holds for $\bar{G}_\rho$ but not necessarily for G_ρ.

If we introduce the notation $\tilde{H} = B^T(V + \bar{\rho}\nabla c^T \nabla c)B$ and $\tilde{g} = B^T d_g$, we can obtain a closed-form solution for this ODE. If this solution is transformed back to the full space, we obtain the trajectory of interest, $\bar{\gamma}(t)$, given by:

$$\bar{\gamma}(t) = B\beta(t) = B\tilde{H}^{-1}\left(\exp(-\tilde{H}t) - I\right)\tilde{g}. \tag{27}$$

Note that the computation of this trajectory requires only the determination of the exponential of a square matrix of dimension two or at most three.

Interior-point methods ensure the positivity of all iterates, to guarantee that the objective function in (2), and in particular its barrier term, is well defined in each iteration. As a consequence, the trajectory defined by $\bar{\gamma}(t)$ must be transformed into another trajectory that lies within the strict interior of the positive orthant. In our algorithm, this is achieved by projecting each infeasible point on the trajectory (27) onto the (perturbed) simple bounds,

$$\hat{\gamma}(t) = \alpha(t)\bar{\gamma}(t), \tag{28}$$

where the scalar $\alpha(t)$ is chosen for each t as

$$\alpha(t) = \min\left\{1, \tau \min\left\{ \frac{x_i}{-\bar{\gamma}_i(t)} \,\middle|\, \bar{\gamma}_i(t) < 0 \right\}\right\}, \tag{29}$$

and τ is defined as in (18).

The next step is to determine an acceptable value for the parameter t in $\hat{\gamma}(t)$. As we will show in Section 4, when $\tilde{H}$ is positive definite we have $\bar{\gamma}(t) \to -B\tilde{H}^{-1}\tilde{g} = d_x$, a reasonable step, as $t \to \infty$. As a consequence, we may need to handle infinite values of the parameter t to determine the next iterate. To avoid the complications associated with these values, the curve is reparametrized so that points of interest, such as this Newton step, can be found by moving a finite distance along the curve projected onto the bounds, $\hat{\gamma}$ (28).

We have chosen to use the following reparametrization, see for instance [1, 21],

$$s = \begin{cases} \dfrac{-1}{\delta_m}\left(e^{-\delta_m t} - 1\right) & \text{if } \delta_m \neq 0, \\ t & \text{if } \delta_m = 0, \end{cases} \tag{30}$$

where $\delta_m \leq \ldots \leq \delta_1$ are the eigenvalues of $\tilde{H}$. Under this reparametrization, if $\delta_m > 0$ then $s \in [0, 1/\delta_m]$.

3.2. Computing the step

Once the trajectory $\hat{\gamma}(s)$ has been computed, we obtain the next iterate in the variables x from an appropriate step along the curve, $x_{k+1} = x_k + \hat{\gamma}(s)$. The value of s is chosen to ensure sufficient descent for our merit function, and it is found by performing a backtracking search starting at $s_0 = 1/\delta_m$. We determine the step s_i as the first value in the sequence $\{s_0/2^i\}_{i=0}^{\infty}$ that satisfies the following sufficient descent condition:

$$L_A(x + \hat{\gamma}(s_i), \bar{\lambda}; \mu, \bar{\rho}) \leq L_A(x, \bar{\lambda}; \mu, \bar{\rho}) - \sigma s_i \min(\|d_g\|^2, \|d_x\|^2). \tag{31}$$

The scalar σ is chosen as a small value $\sigma \in (0, 1)$.

3.3. The multiplier estimates

The value of $\bar{\lambda}$ in (24) should be defined to ensure that the sequence of iterates in the algorithm is associated to a decreasing sequence of values for the merit function, to guarantee the global convergence of the algorithm. This value is kept fixed at all trial points in the trajectory. For the search of the new iterate we define

$$\bar{\lambda} = \begin{cases} \lambda + d_\lambda & \text{if } d_n = 0 \\ \lambda & \text{otherwise,} \end{cases} \tag{32}$$

In practice, this approach may not be satisfactory for all iterations. At the end of the search procedure, the next iterate λ_{k+1} is defined as $\bar{\lambda}$, if there is no negative curvature, the step s_0 is accepted and $\alpha(s_0) \geq 0.95$. Otherwise, we use an approach similar to [16]: the value of λ_{k+1} is chosen as the least-squares estimate at the accepted step.

3.4. Adjusting the penalty parameter

The traditional role of the penalty parameter in a merit function that includes penalty terms, such as (4), is to enforce convergence to points satisfying the constraints $c(x) = 0$. Although the use of the Newton direction generates iterates that satisfy feasibility in the limit, if the penalty parameter is not chosen to be sufficiently large, the Newton direction may not be a descent direction for the merit function and no valid step will be found.

In the proposed algorithm, the combination of directions to define the trajectory $\hat{\gamma}(s)$ automatically ensures that sufficient descent is available at all iterates. As a consequence, the penalty parameter is used to attain other reasonable properties for the search trajectory. In particular, we wish to ensure that the Newton step corresponds to the final point in the trajectory whenever there is no negative curvature in the null-space of the constraints at the current iterate. Note that this may not be true in all cases; a sufficient condition is that the matrix $\tilde{H}$ introduced in (27) is positive definite, as we will show in Section 4.

In the absence of negative curvature, the matrix V introduced in (26) is positive definite in the subspace spanned by W_A. If we assume that ∇c has full row rank then $V + \rho \nabla c^T \nabla c$ is positive definite for large enough ρ. As we wish $\tilde{H}$ to be positive definite, from its definition this condition will hold if and only if $B^T V B + \rho(\nabla c B)^T \nabla c B$ is positive definite. Under the preceding assumptions and the properties of V, this is true for all large enough values of ρ, and we only need to determine a "large enough" value for ρ. From our assumption that there is no direction of negative curvature, the matrix $B^T VB + \rho(\nabla cB)^T \nabla cB$ has dimension 2, and its eigenvalues can be found as the roots of a four-degree polynomial. We determine an acceptable value for ρ, $\bar{\rho}$, such that

$$\lambda_{\min}(B^T VB + \bar{\rho}(\nabla cB)^T \nabla cB) \geq \beta_m, \tag{33}$$

where β_m is a prespecified positive value; if $\lambda_{\min}(B^T VB) \geq \beta_m$, we set $\bar{\rho} = 0$. Finally, we define $\rho_{k+1} = \bar{\rho}$.

4. Properties of the search

In this section we present some basic properties of the procedure to compute the next iterate. Our aim is not to provide any convergence proof for the algorithm; a detailed proof of this sort will be a matter for a different paper. We only wish to establish that the procedure to combine the search directions is well defined, and has reasonable properties regarding the global convergence of an algorithm that uses appropriate search directions and parameter updates.

We will assume that certain properties are satisfied by the functions defining problem (1) and the iterates generated by the algorithm. A global convergence proof should include some of these assumptions and prove that the algorithm satisfies the others.

$A.1$ The iterates x_k generated by the algorithm remain in a compact set, $C \subset \mathbb{R}^n_+$.
$A.2$ The functions f and c have continuous second derivatives in C.
$A.3$ For a given value of the barrier parameter μ_k, the iterates x_k are bounded away from zero, $x_k \geq \beta(\mu_k) > 0$ for some function β.
$A.4$ The multiplier estimates λ remain bounded in norm at all iterates.

Regarding Assumption *A.3*, it can be shown that it holds from the descent properties of the merit function and the logarithmic terms in this function. See [21] and [22] for additional details.

We will ignore the iteration subscript in what follows, whenever the context is clear. We start by establishing that the algorithm is well-defined.

Lemma 1. *At any iteration k, the search described in Section 3 finds a step satisfying (31) in a finite number of iterations.*

Proof. From the continuity of $\bar{\gamma}(t)$, $\bar{\gamma}(0) = 0$ and assumption $A.3$ there exists a value $\bar{t}(\mu_k) > 0$ such that $\alpha(t)$ defined in (29) takes the value one for all $t \in [0, \bar{t}(\mu_k))$. From the reparametrization in (30) and this property, there exists a value $\bar{s}(\mu_k) > 0$ such that $\hat{\gamma}(s) = \bar{\gamma}(s)$ for all $s \in [0, \bar{s}(\mu_k))$. We will only consider these values of s in what follows. We will also omit the iteration subscript k to simplify the notation.

From the definition of the function L_A in (4), the definition of the curve (27), the reparametrization (30) and the Taylor series expansion around $s = 0$ we have

$$\begin{aligned} &L_A(x + \hat{\gamma}(s), \bar{\lambda}; \mu, \bar{\rho}) - L_A(x, \bar{\lambda}; \mu, \bar{\rho}) \\ &\quad = s\nabla_x L_A(x, \bar{\lambda}; \mu, \bar{\rho})^T \frac{d\hat{\gamma}(0)}{ds} + \frac{s^2}{2}\left(\frac{d\hat{\gamma}(0)}{ds}\right)^T \nabla_{xx} L_A(\tilde{x}, \bar{\lambda}; \mu, \bar{\rho}) \frac{d\hat{\gamma}(0)}{ds}, \end{aligned}$$

where $\tilde{x} = x + \zeta\hat{\gamma}(s)$ for some $\zeta \in [0, 1]$ and

$$\frac{d}{ds}\hat{\gamma}(0) = \frac{d}{dt}\bar{\gamma}(0)\frac{dt}{ds} = -B\tilde{g} = -BB^T\nabla_x L_A(x, \bar{\lambda}; \mu, \bar{\rho}).$$

Note that B has columns that form an orthonormal basis for a subspace spanned by $\nabla_x L_A(x, \bar{\lambda}; \mu, \bar{\rho})$ and other directions. This implies $BB^T\nabla_x L_A(x, \bar{\lambda}; \mu, \bar{\rho}) = \nabla_x L_A(x, \bar{\lambda}; \mu, \bar{\rho}) = d_g$. As a consequence,

$$\begin{aligned} &L_A(x + \hat{\gamma}(s), \bar{\lambda}; \mu, \bar{\rho}) - L_A(x, \bar{\lambda}; \mu, \bar{\rho}) + \sigma s\|d_g\|^2 \\ &\quad = -(1-\sigma)s\|d_g\|^2 + \tfrac{1}{2}s^2 d_g^T\nabla_{xx}L_A(\tilde{x}, \bar{\lambda}; \mu, \bar{\rho})d_g \end{aligned}$$

and also,

$$\begin{aligned} &L_A(x + \hat{\gamma}(s), \bar{\lambda}; \mu, \bar{\rho}) - L_A(x, \bar{\lambda}; \mu, \bar{\rho}) + \sigma s\min(\|d_g\|^2, \|d_x\|^2) \\ &\quad \le -(1-\sigma)s\|d_g\|^2 + \tfrac{1}{2}s^2 d_g^T\nabla_{xx}L_A(\tilde{x}, \bar{\lambda}; \mu, \bar{\rho})d_g, \end{aligned} \tag{34}$$

for some value $\tilde{x}$. For sufficiently small s and $\sigma < 1$ the right-hand side is negative and (31) holds. □

To prove global convergence for an algorithm based on this search we should have sufficient descent on the merit function in every iteration, this function should be bounded below and we would also need the value of the parameter s to be bounded away from zero. In Lemma 2, we show that a bound on s that depends on μ can be derived using the same arguments as for the preceding proof. As a consequence, the preceding convergence arguments can be applied for fixed μ.

Lemma 2. *The step s along the curve in each iteration is bounded away from zero by a positive value, $s \ge \hat{s}(\mu) > 0$.*

Proof. From the definition of $\nabla_{xx}L_A$, (6), and Assumptions $A.1$ to $A.4$, it follows that there exists a positive constant $\bar{\beta}$ such that

$$d_g^T\nabla_{xx}L_A(\tilde{x}, \bar{\lambda}; \mu, \bar{\rho})d_g \le \bar{\beta}\|d_g\|^2 + d_g^T M\tilde{X}^{-2}d_g \le \left(\bar{\beta} + \frac{\|\mu\|}{\beta(\mu)^2}\right)\|d_g\|^2.$$

Given this bound, it follows that

$$-(1-\sigma)s\|d_g\|^2 + \tfrac{1}{2}s^2 d_g^T\nabla_{xx}L_A(\tilde{x}, \bar{\lambda}; \mu, \bar{\rho})d_g < 0$$

for all $s \in (0, \bar{s})$, where

$$\bar{s} = \frac{2(1-\sigma)}{\bar{\beta} + \|\mu\|/\beta(\mu)^2}.$$

As a consequence of (34),

$$L_A(x + \hat{\gamma}(s), \bar{\lambda}; \mu, \bar{\rho}) - L_A(x, \bar{\lambda}; \mu, \bar{\rho}) + \sigma s\min(\|d_g\|^2, \|d_x\|^2) < 0,$$

for all $s \in (0, \bar{s})$, and from the backtracking search implemented in the algorithm, the computed step satisfies $s \ge \hat{s} \equiv (1-\sigma)/(\bar{\beta} + \|\mu\|/\beta(\mu)^2)$. □

We prove another result in this section, related to the desirable local convergence properties of the algorithm. We show that if the initial trial value s_0 is accepted, in that iteration we update the variables using the Newton direction. The superlinear convergence of the algorithm must follow from this result if we are able to accept this step and we update μ appropriately. Lemma 3 is an extension of a similar result for unconstrained problems in [1].

Lemma 3. *In those iterations where no negative curvature is used and the multiplier estimate is taken as $\lambda + d_\lambda$, if $\bar{\rho}$ has been chosen as indicated in Section 3.4, we have*

$$\lim_{t\to\infty} \bar{\gamma}(t) = \bar{\gamma}(s_0) = d_x.$$

Proof. From the assumption that no negative curvature is present, the comments in Section 3.1 imply $V = \bar{G}_\rho$, a positive definite matrix, and $\bar{\lambda} = \lambda + d_\lambda$. From (27), as $\bar{\rho}$ has been chosen to ensure that $\tilde{H}$ is positive definite,

$$\lim_{t\to\infty} \bar{\gamma}(t) = \lim_{t\to\infty} B\tilde{H}^{-1}\left(\exp(-\tilde{H}t) - I\right)\tilde{g} = -B\tilde{H}^{-1}\tilde{g}.$$

From (11) we have

$$(V + \bar{\rho}\nabla c^T\nabla c)d_x = -\nabla_x L_A(x, \lambda + d_\lambda; \mu, \bar{\rho}),$$

and using the definitions of $\tilde{H}$ and $\tilde{g}$, as B has columns that form an orthonormal basis for a subspace containing d_x, implying $BB^T d_x = d_x$, we obtain

$$\begin{aligned}\lim_{t\to\infty} \bar{\gamma}(t) &= -B(B^T(V + \bar{\rho}\nabla c^T\nabla c)B)^{-1}B^T\nabla_x L_A(x, \lambda + d_\lambda; \mu, \bar{\rho}) \\ &= B(B^T(V + \bar{\rho}\nabla c^T\nabla c)B)^{-1}B^T(V + \bar{\rho}\nabla c^T\nabla c)d_x \\ &= BB^T d_x = d_x.\end{aligned}$$ □

The barrier parameter update rule (23) is another unconventional part of the algorithm, and the analysis of some of the theoretical properties associated with this rule may be of interest. A similar algorithm, based on a different procedure to combine the search directions but using a similar update rule, was described in [22]. In that reference some global convergence properties were analyzed, and in particular it was shown that under reasonable assumptions $\mu_k \to 0$. In the remainder of this section we study the impact of the update rule on the rate of convergence of the iterates generated by the algorithm, something not analyzed in [22].

We tighten one of the preceding assumptions and introduce additional assumptions on problem (1), to ensure that the directions computed close to a solution are suitable for attaining superlinear convergence.

$A.2m$ The functions f and c have continuous third derivatives in C.
$A.5$ The Jacobian matrix $\nabla c(x)$ has full row rank at all second-order KKT points of problem (1).
$A.6$ Strict complementarity holds at all second-order KKT points of problem (1).
$A.7$ The sufficient optimality conditions hold at all second-order KKT points of the problem.

As we wish to concentrate on the analysis of the impact of the barrier parameter update rule, we will also introduce assumptions on the behavior of other aspects of the algorithm.

$A.8$ The sequence of iterates generated by the algorithm converges to a second-order KKT point of problem (1).
$A.9$ The penalty parameter ρ_k remains bounded in the algorithm.

From these assumptions some convergence results can be derived for the search directions in the algorithm.

Lemma 4. *Under the preceding assumptions, the primal and dual search directions in the algorithm converge to zero.*

Proof. From Assumption *A.8* it holds that

$$\lim_{k\to\infty} \|\nabla f(x_k) - \nabla c(x_k)^T \lambda_k - X_k^{-1}\mu_k\| = 0, \qquad \lim_{k\to\infty} \|c(x_k)\| = 0,$$

and the right-hand side of system (11) converges to zero. By construction of $\bar{G}_\rho$ and Assumption *A.5*, the coefficient matrix of this system is invertible and bounded away from a singular matrix. Thus, it holds that $\lim_{k\to\infty} \|(d_x)_k\| = 0$ and $\lim_{k\to\infty} \|(d_\lambda)_k\| = 0$.

The matrices in the sequence $\{(W_A)_k^T (G_\rho)_k (W_A)_k\}$ are positive definite for all large k, from Assumptions *A.7* and *A.8*, implying that (14) cannot hold and $(d_n)_k = 0$ for large enough iterations k.

Consider the dual update direction $(d_z)_k$ and its definition in (8). From Assumptions *A.7* and *A.8* for all large enough iterations k, $(\bar{G}_\rho)_k = (G_\rho)_k = \nabla_{xx} L(x_k, \lambda_k - \rho_k c(x_k)) + X_k^{-1} Z_k$. From (11),

$$\begin{aligned} X_k^{-1} Z_k (d_x)_k = {} & -\nabla_{xx} L(x_k, \lambda_k - \rho_k c(x_k))(d_x)_k - \nabla f(x_k) \\ & + \nabla c(x_k)^T (\lambda_k + (d_\lambda)_k) + X_k^{-1}\mu_k. \end{aligned} \tag{35}$$

Assumptions *A.7* and *A.8* and $\lim_{k\to\infty}(d_x)_k = 0$ imply that the right-hand side of (35) goes to zero. As a consequence, $\lim_{k\to\infty} \|X_k^{-1} Z_k (d_x)_k\| = 0$. From Assumption *A.8*, it holds that $\lim_{k\to\infty} \|X_k^{-1}\mu_k - z_k\| = 0$. Using these limits in (8) it follows that

$$\lim_{k\to\infty} \|(d_z)_k\| = \lim_{k\to\infty} \|\mu_k - X_k^{-1} z_k - X_k^{-1} Z_k (d_x)_k\| = 0. \qquad \square$$

We next present two results related to the size of the steps selected by the algorithm. The first result shows that the maximum steps that ensure feasibility for the primal and dual variables are eventually arbitrarily close to one in the algorithm. To simplify the notation in what follows, let $\alpha_k \equiv \alpha((s_0)_k)$, the feasibility correction for the initial primal step.

Lemma 5. *There exists an iteration index r such that for all $k \geq r$,*

$$\alpha_k \geq 1 - \|\mu_k\|, \quad (\alpha_d)_k \geq 1 - \|\mu_k\|.$$

Proof. From Assumption *A.8*, let x^* and z^* denote the limit points for the sequences $\{x_k\}$ and $\{z_k\}$. Consider first the bound for α_d and let $\mathcal{I}_z$ denote the set of positive components in z^*, $\mathcal{I}_z = \{i : z_i^* > 0\}$. From Lemma 4, $\lim_{k\to\infty}(d_z)_k = 0$ and there exists an iteration index r_1 such that $|(z_{ki}/(d_z)_{ki}| > 1$ for all $k \geq r_1$ and $i \in \mathcal{I}_z$.

For $i \notin \mathcal{I}_z$, from Assumption *A.6* it holds that $x_i^* > 0$. From (8) it also holds that

$$z_{ki} + (d_z)_{ki} = \frac{z_{ki}}{x_{ki}}\left(\frac{\mu_{ki}}{z_{ki}} - (d_x)_{ki}\right). \tag{36}$$

From Assumption *A.8*, $\mu_{ki}/z_{ki} \to x_i^* > 0$, and from Lemma 4, $\lim_{k\to\infty}(d_x)_k = 0$, implying the existence of an iteration index r_2 such that $\mu_{ki}/z_{ki} - (d_x)_{ki} \geq x_i^*/2 > 0$ and from (36), $z_{ki} + (d_z)_{ki} > 0$ for all $k \geq r_2$ and all $i \notin \mathcal{K}_z$. For these iterations, if $(d_z)_{ki} < 0$ then $-z_{ki}/(d_z)_{ki} > 1$.

Combining the preceding results it holds that $-z_{ki}/(d_z)_{ki} > 1$ for $(d_z)_{ki} < 0$ and all i. Using (17) and (18) we obtain

$$(\alpha_d)_k \geq \tau_k \geq 1 - \|\mu_k\|,$$

for all $k \geq \max(r_1, r_2)$.

Repeating the preceding arguments for the sequence $\{x_k\}$ and using Assumption *A.6* it is straightforward to show that there exists an iteration index r_3 such that

$$\alpha_k \geq \tau_k \geq 1 - \|\mu_k\|, \tag{37}$$

for all $k \geq r_3$. $\square$

Our next result shows that the algorithm eventually accepts the Newton step in all cases, under some conditions.

Lemma 6. *There exists a fixed value $\bar{\sigma} = 0.45\beta_m^2 > 0$ such that if $\sigma \leq \bar{\sigma}$ in (31) then there exists an iteration index r such that for all $k \geq r$ it holds that*

$$x_{k+1} = x_k + \alpha_k(d_x)_k, \quad \lambda_{k+1} = \lambda_k + (d_\lambda)_k.$$

Proof. As a consequence of Lemma 4 there exists an iteration index r_1 such that $(d_n)_k = 0$ for all $k \geq r_1$. Both Lemma 5 and $\lim_{k\to\infty}\mu_k = 0$ imply the existence of an iteration $r_2 \geq r_1$ such that $\alpha_k \geq 0.95$ for all $k \geq r_2$; the update rule in Section 3.3 implies $\lambda_{k+1} = \lambda_k + (d_\lambda)_k$ for all $k \geq r_2$.

For the primal variables, x_k, the desired result follows if condition (31) is satisfied for $s_0 = 1/(\delta_m)_k$ and all large enough values of k. To simplify the notation, let $\Psi_k(x) \equiv L_A(x, \bar{\lambda}_k; \mu_k, \bar{\rho}_k)$. From Lemma 3 for $k \geq r_1$ it holds that $\Psi_k(x_k + \hat{\gamma}_k(s_0)) = \Psi_k(x_k + \alpha_k(d_x)_k)$. From (4) and (16), the Taylor series expansion for this function around $x = x_k$ is given by

$$
\begin{aligned}
&\Psi_k(x_k + \alpha_k (d_x)_k) \\
&= f(x_k + \alpha_k (d_x)_k) - \textstyle\sum_i \mu_{ki} \log(x_{ki} + \alpha_k (d_x)_{ki}) \\
&\quad - \bar{\lambda}_k^T c(x_k + \alpha_k (d_x)_k) + \tfrac{1}{2}\bar{\rho}_k \|c(x_k + \alpha_k (d_x)_k)\|^2 \\
&= \Psi_k(x_k) + \alpha_k \left(\nabla f(x_k) - X_k^{-1}\mu_k - \nabla c(x_k)^T \left(\bar{\lambda}_k - \bar{\rho}_k c(x_k)\right)\right)^T (d_x)_k \\
&\quad + \tfrac{1}{2}\alpha_k^2 (d_x)_k^T \left(\nabla_{xx} L(x_k, \bar{\lambda}_k - \bar{\rho}_k c(x_k)) + M_k X_k^{-2} + \bar{\rho}_k \nabla c(x_k)^T \nabla c(x_k)\right) (d_x)_k \\
&\quad + o(\|(d_x)_k\|^2) \\
&= \Psi_k(x_k) + \alpha_k (d_g)_k^T (d_x)_k + \tfrac{1}{2}\alpha_k^2 (d_x)_k^T \big(\nabla_{xx} L(x_k, \bar{\lambda}_k - \bar{\rho}_k c(x_k)) + M_k X_k^{-2} \\
&\quad + \bar{\rho}_k \nabla c(x_k)^T \nabla c(x_k)\big)(d_x)_k + o(\|(d_x)_k\|^2).
\end{aligned} \tag{38}
$$

Consider terms of the form $(d_x)_{ki}^2(\mu_{ki}/x_{ki} - z_{ki})/x_{ki}$. If $i \in \mathcal{J} \equiv \{i : x_i^* > 0\}$, Assumption *A.8* implies $(d_x)_{ki}^2(\mu_{ki}/x_{ki} - z_{ki})/x_{ki} \le \epsilon (d_x)_{ki}^2$ for any $\epsilon > 0$ and large enough k. If $i \notin \mathcal{J}$, then both terms μ_{ki}/x_{ki}^2 and z_{ki}/x_{ki} converge to infinity and a more detailed analysis is required. Nevertheless, the presence of terms of the form $X_k^{-1}Z_k$ in the coefficient matrix of system (11) implies that the size of the corresponding components $(d_x)_{ki}$ must be very small compared to other components of the search directions. To formalize this statement, from (11),

$$
\begin{aligned}
X_k^{-1}Z_k (d_x)_k = {}& -\nabla f(x_k) + \nabla c(x_k)^T(\lambda_k + (d_\lambda)_k) + X_k^{-1}\mu_k \\
& - \nabla_{xx} L(x_k, \lambda_k - \rho_k c(x_k))(d_x)_k.
\end{aligned}
$$

The definition of the gradient direction $(d_g)_k$ in (16), the definition of $\bar{\lambda}_k$, (32), and Lemma 4 imply

$$
X_k^{-1}Z_k (d_x)_k = -(d_g)_k + \bar{\rho}_k \nabla c(x_k)^T c(x_k) - \nabla_{xx} L(x_k, \lambda_k - \rho_k c(x_k))(d_x)_k.
$$

Using (11) again, it holds that $\bar{\rho}_k \nabla c(x_k)^T c(x_k) = -\bar{\rho}_k \nabla c(x_k)^T \nabla c(x_k)(d_x)_k$, yielding

$$
\begin{aligned}
X_k^{-1}Z_k (d_x)_k = {}& -(d_g)_k - \big(\nabla_{xx} L(x_k, \lambda_k - \rho_k c(x_k)) \\
& + \bar{\rho}_k \nabla c(x_k)^T \nabla c(x_k)\big)(d_x)_k.
\end{aligned} \tag{39}
$$

From Assumption *A.6*, $z_i^* > 0$ for $i \notin \mathcal{J}$ and as a consequence of (39) and Assumptions *A.2*, *A.9*, we have $(d_x)_{ki}/x_{ki} = O(\max(\|(d_x)_k\|, \|(d_g)_k\|))$. Thus, for $i \notin \mathcal{J}$ it holds that $(d_x)_{ki}^2(\mu_{ki}/x_{ki} - z_{ki})/x_{ki} \le o(\|(d_x)_k\| \max(\|(d_x)_k\|, \|(d_g)_k\|))$. Combining the results for the different components i,

$$
(d_x)_k^T \left(M_k X_k^{-2} - X_k^{-1} Z_k\right) (d_x)_k = o\left(\|(d_x)_k\| \max(\|(d_x)_k\|, \|(d_g)_k\|)\right). \tag{40}
$$

From Assumptions *A.2*, *A.8*, *A.9* and (32),

$$
\begin{aligned}
&(d_x)_k^T \left(\nabla_{xx} L(x_k, \bar{\lambda}_k - \bar{\rho}_k c(x_k)) - \nabla_{xx} L(x_k, \lambda_k - \rho_k c(x_k)\right) (d_x)_k \\
&\qquad = o(\|(d_x)\|^2).
\end{aligned} \tag{41}
$$

From the definition of G_ρ, (38), (40) and (41), for large iterations it holds that

$$\Psi_k(x_k+\alpha_k(d_x)_k) = \Psi_k(x_k) + \tfrac{1}{2}\alpha_k^2(d_x)_k^T\left((G_\rho)_k + \bar\rho_k \nabla c(x_k)^T\nabla c(x_k)\right)(d_x)_k \\ + \alpha_k(d_g)_k^T(d_x)_k + o\left(\|(d_x)_k\| \max(\|(d_x)_k\|, \|(d_g)_k\|)\right). \qquad (42)$$

Using (11), for these iterations $\nabla c(x_k)(d_x)_k = -c(x_k)$ and

$$\begin{aligned}(d_x)_k^T(G_\rho)_k(d_x)_k &= -(d_g)_k^T(d_x)_k + \bar\rho_k(d_x)_k^T\nabla c(x_k)^T c(x_k)\\ &= -(d_g)_k^T(d_x)_k - \bar\rho_k c(x_k)^T c(x_k).\end{aligned}$$

Replacing these results in (42) we obtain

$$\Psi_k(x_k+\alpha_k(d_x)_k) = \Psi_k(x_k) + \alpha_k\left(1-\tfrac{1}{2}\alpha_k\right)(d_g)_k^T(d_x)_k \\ + o\left(\|(d_x)_k\|\max(\|(d_x)_k\|, \|(d_g)_k\|)\right). \qquad (43)$$

From (11) and (33) we have

$$(d_g)_k^T(d_x)_k = -(d_x)_k^T((G_\rho)_k + \bar\rho_k\nabla c(x_k)^T\nabla c(x_k))(d_x)_k \qquad (44)$$

$$\Rightarrow (d_g)_k^T(d_x)_k \le -\beta_m\|(d_x)_k\|^2. \qquad (45)$$

Similarly, from (11) and Assumptions *A.2* and *A.9* it holds that

$$(d_g)_k = -((G_\rho)_k + \bar\rho_k\nabla c(x_k)^T\nabla c(x_k))(d_x)_k \Rightarrow \|(d_g)_k\| = O(\|(d_x)_k\|). \qquad (46)$$

From Lemma 5, for all large enough iterations k, $1 \ge \alpha_k \ge 0.995$, implying $\alpha_k(1-0.5\alpha_k) \ge 0.497$. Replacing this bound, (44) and (46) in (43) we obtain

$$\Psi_k(x_k+\alpha_k(d_x)_k) \le \Psi_k(x_k) - 0.497\beta_m\|(d_x)_k\|^2 + o\left(\|(d_x)_k\|^2\right). \qquad (47)$$

From the definition of δ_m and (33) it follows that $s_0 = 1/\delta_m \le 1/\beta_m$. Consider iterations k large enough to satisfy $o\left(\|(d_x)_k\|^2\right) \le 0.047\beta_m\|(d_x)_k\|^2$ and let $\sigma \le 0.45\beta_m^2$, then from (47)

$$\begin{aligned}\Psi_k(x_k+\alpha_k(d_x)_k) &\le \Psi_k(x_k) - 0.45\beta_m\|(d_x)_k\|^2\\ &\le \Psi_k(x_k) - \frac{\sigma}{\beta_m}\|(d_x)_k\|^2\\ &\le \Psi_k(x_k) - \sigma(s_0)_k\|(d_x)_k\|^2\\ &\le \Psi_k(x_k) - \sigma(s_0)_k\min(\|(d_g)_k\|^2, \|(d_x)_k\|^2),\end{aligned}$$

and condition (31) is satisfied, implying that the value $x_k + \alpha_k(d_x)_k$ is accepted as the next iterate. □

We are now able to analyze the local convergence of the iterates generated by the algorithm. We introduce the following notation: let $w = (x^T, \lambda^T, z^T)^T$ denote the set of all primal and dual variables for problems (1) and (2). Also, let w^* denote the corresponding values for a second-order KKT point of problem (1), a limit point of the algorithm from assumption *A.8*. Finally, let $w^*(\mu)$ be a second-order KKT point for problem (2), closest to w^*. Note that under the preceding assumptions for small enough values of μ these KKT points exist and are unique, see [19] for example.

Under the preceding assumptions, we can show the following asymptotic result for the sequence $\{w_k - w^*\}$.

Theorem 1. *For large enough k the sequence of iterates* $\{w_k\}$ *satisfies* $\|w_{k+1} - w^*\| = o(\|w_k - w^*\|)$.

Proof. We can write

$$\|w_{k+1} - w^*\| \le \|w_{k+1} - w^*(\mu_k)\| + \|w^*(\mu_k) - w^*\|. \tag{48}$$

For the second term in the right-hand side of this inequality, using an argument similar to that in [19], the definition of F in (19), $F(w^*) = 0$ and Taylor series expansions yield

$$\begin{pmatrix} 0 \\ 0 \\ \mu_k \end{pmatrix} = F(w^*(\mu_k)) = \nabla F(w^*)(w^*(\mu_k) - w^*) + o(\|w^*(\mu_k) - w^*\|).$$

From assumptions *A.5, A.6* and *A.7* the matrix $\nabla F(w^*)$ is nonsingular, see [19], and as $\mu_k \to 0$ and $w^*(\mu_k) \to w^*$ in the algorithm, we have that

$$\|w^*(\mu_k) - w^*\| = O(\|\mu_k\|). \tag{49}$$

Consider now the term $\|w_{k+1} - w^*(\mu_k)\|$ in (48). From Lemma 4, close enough to the solution we do not have negative curvature on the relevant subspace, $(d_n)_k = 0$, and $(\bar{G}_\rho)_k = \nabla_{xx} L(x_k, \lambda_k - \rho_k c(x_k)) + X_k^{-1} Z_k$. Using Taylor series expansions on the right-hand side of (7) and $F(w^*(\mu_k)) = (0, 0, \mu_k^T)^T$, we have

$$F_k - \begin{pmatrix} 0 \\ 0 \\ \mu_k \end{pmatrix} = \nabla F(w^*(\mu_k))(w_k - w^*(\mu_k)) + o(\|w_k - w^*(\mu_k)\|), \tag{50}$$

where $F_k = F(w_k)$. Note that the higher derivative terms in the preceding expansion are bounded from assumption *A.2m*.

We need a relationship for the iterates w_{k+1}. The left-hand side of (50) can be related to the search directions using their definitions in (11) and (8). From assumption *A.7*, close enough to the solution we have

$$-F_k + \begin{pmatrix} 0 \\ 0 \\ \mu_k \end{pmatrix} = \nabla F_k d_k + \begin{pmatrix} \rho_k \sum_j c_j(x_k) \nabla^2 c_j(x_k) & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} d_k \tag{51}$$

$$= \nabla F_k d_k + Y_k d_k, \tag{52}$$

where $d_k = ((d_x)_k^T, (d_\lambda)_k^T, (d_z)_k^T)^T$ denotes the vector of all search directions and Y_k denotes the required modification of the ∇F_k matrix in (51). From (50) and (51) we obtain

$$\begin{aligned}(\nabla F_k + Y_k)(w_k + d_k - w^*(\mu_k)) + (\nabla F(w^*(\mu_k)) - \nabla F_k - Y_k)(w_k - w^*(\mu_k)) \\ = o(\|w_k - w^*(\mu_k)\|).\end{aligned} \tag{53}$$

From assumption *A.8* we have $c(x_k) \to 0$ and assumption *A.9* then implies $Y_k \to 0$. Using this condition, $w_k - w^*(\mu_k) \to 0$ and assumption *A.2m* it holds that $\|\nabla F_k +$

$Y_k - \nabla F(w^*(\mu_k))\| = o(1)$. As a consequence, for large enough iterations the matrix $\nabla F_k + Y_k$ is invertible and (53) implies

$$w_k + d_k - w^*(\mu_k) = o(\|w_k - w^*(\mu_k)\|). \tag{54}$$

The definition of the iterates in (28) and Section 3.2, as well as (32) and (17), together with Lemmas 5 and 6 and the preceding result yield

$$\begin{aligned} w_{k+1} - w^*(\mu_k) &= w_k + d_k - w^*(\mu_k) + \begin{pmatrix} (1-\alpha_k)(d_x)_k \\ 0 \\ (1-(\alpha_d)_k(d_z)_k \end{pmatrix} \\ &= o(\|w_k - w^*(\mu_k)\|) + o(\|\mu_k\|), \end{aligned} \tag{55}$$

where we have also used (54) and $\|d_k\| = O(\|w_k - w^*(\mu_k)\|)$, a consequence of (51) and the preceding remarks. Thus, from (48), (49) and (55)

$$\begin{aligned} \|w_{k+1} - w^*\| &\le \|w_{k+1} - w^*(\mu_k)\| + \|w^*(\mu_k) - w^*\| \\ &\le o(\|w_k - w^*(\mu_k)\|) + O(\|\mu_k\|) \\ &\le o(\|w_k - w^*\| + \|w^*(\mu_k) - w^*\|) + O(\|\mu_k\|) \\ &= o(\|w_k - w^*\|) + O(\|\mu_k\|). \end{aligned} \tag{56}$$

Given the results in (49) and (55), all that is left is to analyze the relationship between the sizes of μ_k and $w_k - w^*(\mu_k)$. Consider the update of μ_k in (23), and the sizes of μ_k^*, $\hat{\mu}_k$ and θ_k. From (20) and

$$F_k = \nabla F(w^*)(w_k - w^*) + o(\|w_k - w^*\|) \tag{57}$$

we obtain for large enough k,

$$\theta_k = \|F_k\|^2 = O(\|w_k - w^*\|^2).$$

From the definition of $\hat{\mu}_k$ in Section 2.2.2, the fact that the vector $X_k z_k$ is a subset of the components in F_k and (57) we have

$$\hat{\mu}_k = e^T X_k z_k / n = O(\|w_k - w^*\|).$$

Finally, from the definition of μ_k^* in Section 2.2.2 we have

$$\mu_k^* = \theta_k (X_k z_k) / \|X_k z_k\|^2. \tag{58}$$

To derive an upper bound for this value we need a lower bound for its denominator. Consider first those components such that $(x_i)_k \to x_i^* > 0$ and $(z_i)_k \to 0$. For any $0 < \delta \ll (x_i)^*$ and all large enough iterations we have $|x_i - x_i^*| \le \delta$ and $|x_i + (d_x)_i - x_i^*| \le \delta$. The definition (8) then implies (we omit the iteration subscript to simplify notation)

$$\begin{aligned} \mu_i &= z_i(x_i + (d_x)_i) + x_i(d_z)_i \\ &= x_i^*(z_i + (d_z)_i) + z_i(x_i + (d_x)_i - x_i^*) + (d_z)_i(x_i - x_i^*) \\ &\le (x_i^* + \delta)(z_i + (d_z)_i) \end{aligned}$$

and also

$$(x_i + (d_x)_i)(z_i + (d_z)_i) \ge (x_i^* - \delta)(z_i + (d_z)_i) \ge \mu_i \frac{x_i^* - \delta}{x_i^* + \delta} = (1-\epsilon)\mu_i,$$

where $\epsilon = 2\delta/(x_i^* + \delta)$ can be chosen to be arbitrarily small. Note that, using the symmetry in (8) between x and z, we can apply a similar argument to those components such that $z_i \to z_i^* > 0$ and $x_i \to 0$ and derive the corresponding bound. Assumption *A.6* implies that no other cases are possible. As a consequence of this bound and Lemma 5 we have

$$\begin{aligned} X_k z_k &= (X_{k-1} + (D_x)_{k-1})(z_{k-1} + (d_z)_{k-1}) - (1-\alpha_{k-1})(D_x)_{k-1} z_{k-1} \\ &\quad - (1-(\alpha_d)_{k-1}) X_{k-1} (d_z)_{k-1} \\ &\ge (1-\epsilon)\mu_{k-1} + o(\|\mu_{k-1}\|) \ge (1-\bar\epsilon)\mu_{k-1}, \end{aligned}$$

for some small positive constant $\bar\epsilon$, where $D_x = \operatorname{diag}(d_x)$ and we have also used $(d_x)_{k-1} = o(1)$, $(d_z)_{k-1} = o(1)$ from Lemma 4. Using this bound in (58) we obtain

$$\mu_k^* = O(\|w_k - w^*\|^3/\|\mu_{k-1}\|^2).$$

As a consequence of (23) and the preceding bounds, we have that

$$\|\mu_k\| \le K_1 \exp(-K_2\|w_k - w^*\|^{-2})\|w_k - w^*\| \max(1, \|w_k - w^*\|^2/\|\mu_{k-1}\|^2), \tag{59}$$

for some positive constants K_1, K_2.

Consider two cases regarding the size of μ_{k-1}:

(i) If $\|\mu_{k-1}\| \ge L\exp(-K_2\|w_k - w^*\|^{-2}/2)\|w_k - w^*\|^{-\delta/2}$ for some positive constants L and $\delta \le 1$, then

$$K_1 \exp(-K_2\|w_k - w^*\|^{-2})\|w_k - w^*\| \le (K_1/L^2)\|w_k - w^*\|^{1+\delta}\|\mu_{k-1}\|^2.$$

From this bound and (59) it follows for large k that

$$\begin{aligned} \|\mu_k\| &\le (K_1/L^2)\|w_k - w^*\|^{1+\delta} \max(\|\mu_{k-1}\|^2, \|w_k - w^*\|^2) \\ &\le (K_1/L^2)\|w_k - w^*\|^{1+\delta}, \end{aligned}$$

as from Assumption *A.8* it holds that $\max(\|\mu_{k-1}\|^2, \|w_k - w^*\|^2) \le 1$ for all large enough iterations k. Using (56), for some positive constant K_3 and large enough k,

$$\begin{aligned} \|w_{k+1} - w^*\| &\le K_3\|\mu_k\| + o(\|w_k - w^*\|) \\ &\le (K_1K_3/L^2)\|w_k - w^*\|^{1+\delta} + o(\|w_k - w^*\|) = o(\|w_k - w^*\|). \end{aligned}$$

(ii) Otherwise, if $\|\mu_{k-1}\| < L\exp(-K_2\|w_k - w^*\|^{-2}/2)\|w_k - w^*\|^{-\delta/2}$, as (23) implies $\|\mu_k\| \le O(\|\mu_{k-1}\|)$ and for large enough k and any positive constant $\tilde K_1$ we have

$$\exp(-K_2\|w_k - w^*\|^{-2}/2) \le \tilde K_1\|w_k - w^*\|^{1+3\delta/2},$$

we obtain $\|\mu_k\| \le \tilde K_2\|w_k - w^*\|^{1+\delta}$ for some constant $\tilde K_2$. Equation (56) yields again the desired result. □

5. Implementation and numerical results

5.1. The algorithm

We present a detailed scheme of the proposed interior point algorithm (**G**radient **F**low **I**nterior **P**oint **M**ethod - **GFIPM**), summarizing those aspects described in the previous sections.

```
Algorithm GFIPM

Choose initial values for $x_0$, $\lambda_0$ and $\sigma_0$.
Choose initial values for the scalar $\rho_0$ and the vector $\mu_0$
Set $k = 0$
repeat
    Compute $d_x$ and $d_\lambda$ from (11) using the factorization described
        in [15], and $d_z$ from (8)
    Compute, if it exists, $d_n$, a direction of negative curvature
        from (15)
    Set $d_n = 0$ if (14) is not satisfied
    Compute $\bar{\rho}$ from the procedure in 3.4
    Compute $\bar{\lambda}$ from (32)
    Compute $s$ using a backtracking search until (31) is satisfied
    $x_{k+1} = x_k + \hat{\gamma}(s)$
    Update $\lambda_{k+1}$ from $\lambda_k$ and $d_\lambda$ using the procedure in 3.3
    Compute $\alpha_d$ from (17)
    $z_{k+1} = z_k + \alpha_d d_z$
    Compute the updated barrier vector $\mu_{k+1}$ from (23)
    $\rho_{k+1} = \bar{\rho}$
    $k = k + 1$
until convergence
```

5.2. Numerical results

We have conducted a set numerical experiments on a collection of test problems using algorithm GFIPM. The algorithm has been implemented and the tests have been carried out in MATLAB. The test set we have considered is composed of 150 small problems from the CUTE collection, see Bongartz et al. [3], selected from those nonlinear constrained problems having less than 100 variables and continuous derivatives (note that exact first and second derivatives have been used). This test is the one used in [22], with the addition of some convex quadratic problems. The algorithm has been implemented to include both lower and upper bounds in the barrier terms.

Whenever possible, the initial points given in CUTE have been used. Sometimes these initial points do not satisfy the bound constraints. Such points have been transformed following an automatic strategy similar to that described in Vanderbei and Shanno

[25]. Table 1 shows the results obtained by GFIPM for these problems. The termination criterion used has been

$$\|F(x, \lambda, z; \rho)\| \leq \epsilon(1 + \|\nabla f(x)\|),$$

where $\epsilon = 10^{-8}$, except for problems DISC2 and HS91, where $\epsilon = 10^{-7}$ (for these two problems the algorithm did not converge when $\epsilon = 10^{-8}$ was used).

Table 1. Results for small-size problems

Prob.	Obj.	Const.	KKT	Iter.	Eval.	NC
AIRPORT	47952.7017	3.1e-14	9.9e-12	15	15	0
ALJAZZAF	75.005	2.5e-09	8.9e-07	20	37	0
ALSOTAME	0.08208499	0	7.1e-11	8	8	0
BIGGSC4	-24.499999	1.5e-15	5.2e-08	21	26	2
	-24.5	1.8e-15	1.9e-15	21	21	0
BT13	0	1.3e-08	1.3e-08	20	26	0
CANTILVR	1.33995636	3.1e-11	3.3e-11	16	58	0
CB2	1.95222449	6.9e-12	7.6e-12	11	14	0
CB3	2.0	2.5e-12	2.5e-12	10	23	0
CHACONN1	1.95222449	1.6e-11	2.8e-11	8	9	0
CHACONN2	2.0	2.5e-11	4.3e-11	10	11	0
CONGIGMZ	28.0	8.5e-12	8.9e-12	21	34	0
CSFI1	-49.0752	1.3e-10	1.5e-09	11	13	0
CSFI2	55.0176056	1.2e-13	1.7e-13	14	17	0
DEMYMALO	-3.0	2.3e-11	2.5e-11	11	13	0
DIPIGRI	680.63006	1.6e-08	3.6e-08	11	26	1
	680.63006	1.7e-11	4.4e-11	12	18	0
DISC2	1.5624999	2.4e-08	2.4e-08	63	208	0
DUAL1	0.035012968	1.9e-16	7.0e-12	21	21	0
DUAL2	0.033733671	6.1e-16	9.0e-09	13	13	0
DUAL4	0.746090649	2.1e-16	2.3e-08	14	14	0
EXPFITA	0.0011366117	1.8e-14	1.0e-09	32	32	0
FCCU	11.14910914	4.4e-15	4.2e-14	8	8	0
GIGOMEZ1	-3.0	1.9e-14	2.0e-14	11	20	0
HATFLDH	-24.5	2.7e-15	3.5e-15	14	20	1
	-24.5	2.9e-15	3.9e-15	13	14	0
HIMMELBI	-1735.569579	8.0e-14	3.2e-11	29	29	0
HIMMELBK	0.0518143	3.5e-12	3.5e-12	18	18	0
HIMMELP2	-8.19803189	3.8e-14	3.8e-14	11	15	0
HIMMELP3	-59.0131239	5.2e-10	4.9e-09	8	23	0
HIMMELP4	-59.0131239	9.7e-13	9.8e-13	11	12	0
HIMMELP5	-59.0131239	3.5e-09	3.6e-09	44	148	0
HIMMELP6	-59.0131239	5.7e-12	5.9e-12	16	39	0
HONG	22.57108736	0	6.4e-13	7	7	0
HS10	-0.9999999	1.4e-08	1.4e-08	13	46	0
HS11	-8.49846422	1.4e-14	2.4e-14	7	7	0
HS12	-30.0	1.2e-08	1.2e-08	8	8	0
HS13	--	--	--	--	--	--
HS14	1.39346498	5.1e-12	2.2e-11	9	38	0
HS15	306.50	8.1e-13	3.1e-10	16	34	0
HS16	0.25	1.1e-16	2.5e-16	13	14	0

Table 1. (cont.)

Prob.	Obj.	Const.	KKT	Iter.	Eval.	NC
HS17	1.0	1.7e-12	5.5e-10	17	79	0
HS18	5.0	0	5.6e-17	28	188	0
HS19	-6961.81388	2.1e-11	2.0e-08	14	18	0
HS20	40.19873021	2.1e-09	1.6e-06	6	16	0
HS21	-99.9599999	3.6e-15	2.9e-14	5	5	0
HS21MOD	-99.9599999	0	9.7e-16	11	11	0
HS22	1.0	3.3e-09	1.5e-08	5	5	0
HS23	2.0	1.8e-12	1.8e-12	8	9	0
HS24	-4.0e-97	0	6.4e-19	13	31	1
	-1.0	7.1e-19	7.0e-11	6	6	0
HS29	-22.62741699	7.4e-10	8.7e-10	7	8	0
HS30	1.0	2.4e-09	5.0e-09	5	5	0
HS31	5.999999	9.4e-12	3.9e-09	5	5	0
HS32	1.0	7.8e-11	4.5e-10	8	9	0
HS33	-4.5857864	3.2e-11	3.2e-11	8	8	0
HS34	-0.83403244	4.3e-12	4.3e-12	8	8	0
HS35	0.11111111	1.1e-17	1.9e-10	7	7	0
HS36	-3299.9999	3.5e-15	9.8e-12	8	8	1
	-3299.9999	2.9e-27	9.7e-12	8	8	0
HS37	-3456	2.8e-21	9.7e-14	6	6	0
HS41	1.92592592	0	1.4e-12	7	7	0
HS43	-44.0	6.5e-12	3.2e-11	9	9	0
HS44	-13.0	1.0e-15	2.2e-15	9	9	0
HS44NEW	-13.0	1.0e-15	2.2e-15	9	9	0
HS53	4.0930232	1.8e-15	6.2e-14	4	4	0
HS57	0.0306476	5.9e-10	6.7e-10	20	50	7
	0.0306476	5.6e-10	6.4e-10	25	50	0
HS59	-6.749505	1.0e-14	1.2e-14	66	202	0
HS60	0.03256682	6.5e-12	1.8e-11	7	7	0
HS63	961.7151721	1.2e-08	2.4e-08	6	9	0
HS64	6299.84243	1.1e-16	9.1e-13	17	22	0
HS65	0.95352886	3.5e-15	4.4e-15	10	14	1
	0.95352886	7.1e-15	7.2e-15	11	12	0
HS66	0.518163274	5.1e-15	5.3e-15	10	10	0
HS67	-1162.119226	2.3e-12	2.3e-12	8	10	0
HS68	-0.920425	1.2e-16	1.2e-14	27	72	0
HS69	-956.712887	1.1e-10	1.6e-07	12	12	0
HS70	0.1870436431	2.9e-11	1.2e-09	22	39	0
HS71	17.0140173	4.1e-08	4.1e-08	8	8	0
HS72	727.67936	3.4e-16	1.2e-13	22	42	0
HS73	29.894378	1.0e-08	1.1e-08	11	11	0
HS74	5126.4981	1.4e-12	4.6e-10	8	8	0
HS75	5174.4127	6.8e-13	2.0e-09	8	8	0

Table 1. (cont.)

Prob.	Obj.	Const.	KKT	Iter.	Eval.	NC
HS76	-4.681818181	8.3e-16	1.3e-09	7	7	0
HS80	0.0539498	3.0e-10	3.1e-10	7	9	0
HS81	0.0539498	1.5e-10	1.5e-10	8	8	0
HS83	-30665.539	3.2e-14	7.7e-13	18	18	0
HS84	-5280335.13	5.6e-08	1.6e-04	19	23	0
HS86	-32.348679	1.0e-14	3.6e-08	14	14	0
HS88	1.362656815	3.2e-14	5.0e-10	24	314	0
HS91	1.36265681	4.4e-11	4.8e-08	18	167	0
HS92	1.3626568	4.2e-13	1.4e-07	20	39	5
	1.3626568	5.0e-14	2.6e-08	27	32	0
HS93	135.075963	3.2e-15	1.5e-07	9	9	0
HS95	0.0156195	3.4e-12	3.4e-12	11	11	0
HS96	0.0156195	1.7e-12	1.7e-12	11	11	0
HS97	4.0712463	2.2e-10	4.4e-08	12	34	0
HS98	4.0712463	6.8e-14	6.7e-11	15	33	0
HS99	-8.3108e+08	2.9e-11	0.49945443	6	6	0
HS100	680.630057	1.6e-08	3.6e-08	11	26	1
	680.630057	1.7e-11	4.4e-11	12	18	0
HS104	3.9511634	4.4e-10	2.4e-09	9	12	0
HS105	1044.725129	2.0e-17	1.1e-10	16	19	1
	1044.725129	2.6e-18	5.0e-11	16	19	0
HS106	7049.24802	3.9e-10	3.9e-10	10	54	0
HS107	4797.98188	2.6e-10	1.0e-05	10	78	0
HS108	-0.6749814	1.5e-14	1.9e-14	12	15	0
HS109	5362.06918	4.8e-08	4.9e-08	12	32	0
HS110	-45.7784697	--	4.8e-13	5	5	0
HS111	-47.7610913	2.7e-08	4.9e-08	12	32	0
HS112	-47.7610908	2.5e-06	1.8e-08	11	11	0
HS113	24.306209	1.6e-11	2.9e-11	33	55	0
HS114	-1768.80696	2.1e-11	7.9e-11	16	16	0
HS116	97.5875096	5.6e-09	7.6e-09	33	40	0
HS117	32.3486790	3.4e-10	1.0e-09	17	19	0
HS118	664.820450	2.1e-14	1.1e-12	14	14	0
HS119	244.899697	6.3e-16	2.9e-07	11	11	0
HS268	4.9e-9	9.7e-15	9.8e-09	17	19	0
HUBFIT	0.016893495	2.9e-17	2.8e-09	7	7	0
KIWCRESC	1.2e-09	3.3e-09	3.8e-09	11	16	0
LAUNCH	9.004903149	6.8e-08	6.8e-08	22	24	1
	9.004903149	5.1e-10	3.8e-07	15	15	0
LIN	-0.020198312	4.4e-17	7.5e-15	15	16	0
LOADBAL	0.4528510391	1.3e-13	2.0e-10	13	13	0
MADSEN	0.616432435	9.7e-12	4.7e-11	15	33	0
MAKELA1	-1.414213564	1.1e-13	1.8e-11	19	24	0
MAKELA2	7.1999999	6.4e-11	8.5e-11	7	7	0

Table 1. (cont.)

Prob.	Obj.	Const.	KKT	Iter.	Eval.	NC
MAKELA3	0	2.0e-10	2.1e-10	15	22	0
MATRIX2	0	8.0e-13	9.2e-09	26	27	2
	1.7e-31	9.1e-19	5.7e-12	39	45	0
MIFFLIN1	-1.0	3.2e-09	1.5e-08	5	5	0
MIFFLIN2	-0.9999999	1.4e-10	1.8e-10	13	28	0
MINMAXBD	115.7064397	6.4e-11	6.4e-11	25	36	0
MINMAXRB	3.5e-17	1.3e-11	1.3e-11	7	13	0
MISTAKE	-1.0	5.6e-09	6.3e-09	10	10	0
ODFITS	-2380.026775	8.0e-13	8.4e-13	8	8	0
POLAK1	2.718281833	3.5e-14	6.1e-14	8	8	0
POLAK2	54.59815	1.7e-09	2.1e-09	16	25	0
POLAK3	5.9330033	2.2e-09	4.7e-09	15	50	0
POLAK4	6.0e-17	3.9e-14	3.9e-14	20	22	0
POLAK5	49.99999	1.4e-08	1.7e-08	6	6	1
	49.99999	2.0e-08	2.0e-08	47	69	0
POLAK6	-44	2.9e-09	2.6e-09	14	32	0
PRODPL0	58.79009997	2.3e-09	4.5e-08	16	16	0
PRODPL1	35.73896744	2.4e-12	1.7e-11	23	37	2
	35.73896744	2.1e-14	3.4e-13	14	16	0
QPCBLEND	-0.0078425	5.3e-15	4.1e-11	43	43	0
QPNBLEND	-0.00913614	1.4e-14	5.8e-08	23	23	0
RK23	0.8333333	2.3e-10	3.4e-09	7	7	0
ROSENMMX	-44.0	1.7e-13	1.7e-13	31	87	0
S268	4.9e-09	9.7e-15	9.9e-15	17	19	0
TAME	3.1e-33	0	2.0e-15	3	4	0
TENBARS4	368.4931619	9.9e-12	9.9e-12	15	18	0
TRUSPYR1	11.22874087	2.4e-12	1.1e-11	9	9	0
TRUSPYR2	11.22874090	5.9e-12	6.0e-12	12	12	0
TRY-B	1.2e-25	1.9e-10	1.9e-10	10	11	0
TWOBARS	1.508652417	2.3e-15	4.0e-15	13	170	0
WOMFLET	6.6e-13	5.6e-10	5.6e-10	11	23	0
ZECEVIC2	-4.125	6.3e-16	9.0e-09	8	40	0
ZECEVIC3	97.30945002	7.0e-09	3.1e-08	10	10	0
ZECEVIC4	7.557507769	4.8e-16	7.3e-15	9	13	0
ZY2	2.0	3.8e-09	3.8e-09	6	6	0

The columns in the table correspond to:

- `Prob.`: problem name.
- `Obj.`: value of the objective function, $f(x)$, at the solution.
- `Const.`: norm of the constraint vector, $\|c(x)\|$, at the solution, including slacks.
- `KKT`: norm of the first-order KKT conditions at the solution, $\|F(x, \lambda, z; \rho)\|$.
- `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 in which directions of negative curvature were used.

In those cases where negative curvature was detected the problem was solved a second time, setting the direction of negative curvature to zero. Table 1 includes two lines for those problems, one for the results from each of the two versions of the algorithm.

5.3. *Analysis of the results*

The algorithm was able to solve all problems but one, problem HS13 (that does not satisfy a constraint qualification at the solution). For some of the problems the code finds better local minimizers than those given in [20] (this happened for problems HS105, HS106, HS107, HS112 and HS116), while for other problems these local minimizers are worse (HS59, HS70, HS97, HS98 and HS108). 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.

In general, the number of iterations required to solve the problems is fairly small. The number of function evaluations is higher, but no particular care was taken when implementing a strategy to find a value of the parameter s that satisfied (31); a standard backtracking search was used. It is also interesting to note the large number of cases in which a step s_0 (the equivalent to a unit step) was accepted.

Table 2 presents a brief summary of the results, both iteration counts and function evaluations, for all problems that make use of negative curvature, as well as the size of these problems. The last four columns in this table correspond to:

– `Iter.nc`: iteration count (with negative curvature enabled).
– `Iter`: iteration count (with negative curvature disabled).
– `Eval.nc`: number of function evaluations (with negative curvature enabled).
– `Eval`: number of function evaluations (with negative curvature disabled).

The first part of the table (the first eleven problems) corresponds to the cases where the algorithm used negative curvature when started from the standard initial point (the one specified in CUTE). The second part of the table corresponds to problems where the algorithm in [22] used negative curvature. When the algorithm described in this paper was given the standard initial point for these problems, it made no use of this negative curvature. This is due to the fact that the gradient of the merit function usually has a significant amount of negative curvature information; also, the updating rule for the penalty parameter is more demanding than other proposals based on descent conditions for the merit function, resulting in more frequent updates that force the direction of negative curvature to be discarded. To obtain additional results taking advantage of directions of negative curvature, alternative initial points were introduced for these problems. These initial points were chosen to ensure that negative curvature was detected and used by the algorithm, at least in the first iteration.

For the whole test set and the standard initial points, negative curvature was used in only 7% of the cases. In [22], where a standard line search procedure is used (without any gradient information), negative curvature was used for 23% of the problems in a similar test set. Note that the use of the gradient incorporates negative curvature information in

Table 2. Problems using directions of negative curvature

Prob.	Var.	Cons.	Iter.nc	Iter.	Eval.nc	Eval.
HS24	2	3	13	6	31	6
HS36	3	1	8	8	8	8
HS65	3	1	10	11	14	12
POLAK5	3	2	6	47	6	69
BIGGSC4	4	7	21	21	26	21
HATFLDH	4	7	14	13	20	14
DIPIGRI	7	4	11	12	26	18
HS100	7	4	11	12	26	18
HS105	8	1	16	16	19	19
LAUNCH	25	28	22	15	24	15
PRODPL1	60	29	23	14	37	16
CSFI1	5	4	7	8	8	8
EXPFITA	5	22	18	19	22	19
HIMMELP2	2	1	13	14	27	15
HIMMELP3	2	2	8	10	30	15
HIMMELP4	2	3	13	12	21	14
HIMMELP5	2	3	10	10	11	10
HIMMELP6	2	5	13	13	50	58
HS44	4	6	6	8	7	8
HS44NEW	4	6	6	8	7	8
HS59	2	3	10	37	80	555
HS113	10	8	21	21	31	28
LIN	4	2	6	6	6	6
MISTAKE	9	13	17	19	27	28
POLAK4	3	3	15	17	68	31
ROSENMMX	5	4	29	85	32	68
WOMFLET	3	3	16	17	25	24
ZEZEVIC3	2	2	9	9	9	18
TOTAL			372	488	698	1129
AVERAGE			13.3	17.4	24.9	40.3

an implicit manner. For the whole set of 28 problems where the algorithm used negative curvature, an average reduction of 4.1 iterations was observed (compared to the case where the negative curvature was disabled).

The preceding table also includes a certain number of cases in which using negative curvature was worse than ignoring it. Globally, the reductions in iterations and function evaluations seem to be more significant than the increases. The largest deterioration in the number of iterations amounted to 9 iterations (39%) for problem PRODPL1 and 25 function evaluations (56%) for problem HS24, while the largest improvement was 41 iterations (87%) and 63 function evaluations (91%) for problem POLAK5. Nevertheless, from the observation of the different behavior in the numbers of iterations and function evaluations, special care should be taken when computing the parameter s in the search, in order to reduce the number of function evaluations whenever negative curvature is used. For example, a procedure based on polynomial models for the univariate search would be likely to contribute to the improvement in the behavior of the algorithm.

Table 3 compares the results from the proposed algorithm (with and without negative curvature) to those of other codes reported in the literature, in particular those from [25],

Table 3. Iteration counts for different nonlinear interior point codes

Prob.	GF+nc	GF	VS	Y	GOW
HS64	17	17	28	29	32.5
HS65	10	11	14	15	9
HS71	8	8	12	8	16.5
HS72	22	22	21	43	11
HS73	11	11	20	12	11
HS83	18	18	15	16	14
HS84	19	19	18	21	25
HS93	9	9	10	29	17
HS95	11	11	18	13	20
HS96	11	11	22	12	20.5
HS97	12	12	18	22	31
HS98	15	15	19	20	27
HS100	11	12	11	16	10
HS104	9	9	14	19	12
HS106	10	10	33	39	45
HS108	12	12	23	62	13.5
HS109	12	12	49	21	32
HS113	33	33	16	25	13
HS114	16	16	31	47	15
HS116	33	33	33	82	--
HS117	17	17	22	36	33
HS118	14	14	17	34	17
Average	15.0	15.1	21.1	28.2	20.2

[28] and [16], on a set of 22 HS problems (all the problems that were reported in all of the references). The columns in the table correspond to the number of iterations (matrix factorizations) required by:

– GF+nc: the proposed algorithm, using negative curvature.
– GF: the proposed algorithm, when negative curvature was disabled.
– VS: iteration counts for LoQo, as reported in [25].
– Y: iteration counts reported in [28].
– GOW: iteration counts reported in [16].

From these results the proposed algorithm works better on the average than any of the other three codes. Note that none of the three algorithms uses negative curvature explicitly. All initial points for the algorithms are those indicated in [20]. For the GOW algorithm, the noninteger results represent the average for all the starting points given in [16].

6. Conclusions

We have described a procedure to combine different search directions, including directions of negative curvature if they exist, and an algorithm to solve general nonlinear optimization problems, based on a primal-dual approach, that uses the combination procedure. A local convergence analysis has also been conducted to prove that the rate of convergence of the algorithm using the proposed barrier parameter update is superlinear.

The algorithm has been shown to be efficient on a set of small test problems. The combination of the directions is also very efficient, as shown in the reduced number of iterations required by the algorithm. A clear advantage is that the scaling of the different directions is done in a natural way.

Although this procedure has been applied to cases in which we had either two or three directions to combine, it would be straightforward to extend it to additional directions, such as for example additional directions of negative curvature if they are available.

The impact of the negative curvature is not very significant on these small problems (it is used in only 7% of them), due to the use of the gradient in the search, but it can be quite important in some cases. If the use of negative curvature is forced by choosing an appropriate initial point, the decrease in the iteration count is more significant. Given the limited cost of computing a direction of negative curvature whenever an appropriate factorization is used to obtain the movement directions, it is reasonable for nonconvex problems to take into account this second-order information.

The proposed procedure would also be able to take into account other directions than the ones used in this paper. This alternative would have particular interest in the case of large-scale problems, as the computational cost of conjugate-gradient based directions might be significantly smaller than that of the Newton direction. Further research would be needed to determine the best combination of directions to use in these cases.

Acknowledgements. We wish to thank an anonymous referee for his/her numerous suggestions that have substantially improved the paper.

References

1. Behrman, W.: An efficient gradient flow method for unconstrained optimization. Ph.D. Thesis, Stanford University, June 1998
2. Bertsekas, D.P.: *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York, 1982
3. Bongartz, I., Conn, A.R , Gould, N.I.M., Toint, Ph.L.: CUTE: Constrained and Unconstrained Testing Environment. Trans. ACM Math. Software **21**, 123–160 (1995)
4. Botsaris, C.A.: A curvilinear optimization method based upon iterative estimation of the eigensystem of the Hessian matrix. J. Math. Anal. Appl. **63**, 396–411 (1978)
5. Bunch, J.R., Kaufman, L., Parlett, B.N.: Decomposition of a symmetric matrix. Numer. Math. **27**, 95–109 (1976)
6. Byrd, R.H., Schnabel, R.B., Shultz, G.A.: Approximate solution of the trust region problem by minimization over two-dimensional subspaces. Math. Program. **40**, 247–263 (1988)
7. Conn, A.R., Gould, N., Toint, Ph.L.: A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. SIAM J. Numer. Anal. **28**, 545–572 (1991)
8. Courant, R.: Variational methods for the solution of problems of equilibrium and vibrations. Bull. Amer. Math. Soc. **49**, 1–23 (1943)
9. Del Gatto, A.: A subspace method based on a differential equation approach to solve unconstrained optimization problems. Ph.D. Thesis, Stanford University, June 2000
10. Dennis, J.E., Schnabel, R.B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice Hall: Englewood Cliffs, 1983
11. El-Bakry, A.S., Tapia, R.A., Tsuchiya, T., Zhang, Y.: On the formulation and theory of the Newton interior-point method for nonlinear programming. J. Optim. Theory Appl. **89**, 507–541 (1996)
12. Evtushenko, Y.E., Zhadan, V.G.: Stable barrier-projection and barrier-Newton methods in nonlinear programming. In: *Optimization Methods and Software,* Vol. 3, pp. 237–256, 1994

13. Fiacco, A.V., 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)
14. Fletcher, R.: *Practical Methods of Optimization.* John Wiley, Chichester, 1991
15. Forsgren, A., Murray, W.: Newton methods for large-scale linear equality-constrained minimization. SIAM J. Matrix Anal. Appl. **14**, 560–587 (1993)
16. Gay, D.M., Overton, M.L., Wright, M.H.: A primal-dual interior method for nonconvex nonlinear programming. In: Y. Yuan (ed.), *Advances in Nonlinear Programming,* pp. 31–56. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998
17. Gill, P.E., Murray, W., Saunders, M.A., 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
18. Gill, P.E., Murray, W., Wright, M.H.: *Practical Optimization.* Academic Press, London/New York, 1981
19. Gould, N.I.M., Orban, D., Sartenaer, A., Toint, Ph.L.: Superlinear convergence of primal-dual interior point algorithms for nonlinear programming. SIAM J. Optim. **11**(4), 974–1002 (2001)
20. Hock, W., Schittkowski, K.: *Test Examples for Nonlinear Programming Codes.* Springer Verlag. Berlin, 1981
21. Moguerza, J.M.: Interior point methods for nonconvex optimization. Ph.D. Thesis, Department of Statistics and Econometrics, Universidad Carlos III de Madrid. May, 2000 (in Spanish)
22. Moguerza, J.M., Prieto, F.J.: An augmented Lagrangian interior-point method using directions of negative curvature. Working Paper 00-36. Department of Statistics and Econometrics, Universidad Carlos III de Madrid, 2000
23. Moré, J.J., Sorensen, D.C.: On the use of directions of negative curvature in a modified Newton method. Math. Program. **16**, 1–20 (1979)
24. Schropp, J.: Using dynamical systems to solve minimization problems. Appl. Numer. Math. **18**, 321–335 (1995)
25. Vanderbei, R.J., Shanno, D.F.: An interior-point algorithm for nonconvex nonlinear programming. Comput. Optim. Appl. **13**, 231–252 (1999)
26. Wright, M.H.: Interior methods for constrained optimization. In: A. Iserles (ed.), *Acta Numerica 1992*, Cambridge University Press, N.Y., pp. 341–402, 1992
27. Wright, M.H.: Ill-conditioning and computational error in primal-dual methods for nonlinear programming. SIAM J. Optim. **9**, 84–111 (1998)
28. Yamashita, H.: A globally convergent primal-dual interior point method for constrained optimization. Optim. Meth. Software **10**, 443–469 (1998)
29. Zang, I.: A new arc algorithm for unconstrained optimization. Math. Program. **15**, 36–52 (1978)