Some iterative methods for the solution of a symmetric indefinite KKT system [*]

Silvia Bonettini[1], Valeria Ruggiero[2]
[1] Dipartimento di Matematica, Università di Modena e Reggio Emilia
[2] Dipartimento di Matematica, Università di Ferrara

Abstract

This paper is concerned with the numerical solution of a Karush–Kuhn–Tucker system. Such symmetric indefinite system arises when we solve a nonlinear programming problem by an Interior–Point (IP) approach. In this framework, we discuss the effectiveness of two inner iterative solvers: the method of multipliers and the preconditioned conjugate gradient method. We discuss the implementation details of these algorithms in an IP scheme and we report the results of a numerical comparison on a set of large scale test–problems arising from the discretization of elliptic control problems.

Keywords: Indefinite symmetric KKT system, large scale nonlinear programming problems, Interior–Point method, Hestenes multipliers' method, preconditioned conjugate gradient method, elliptic control problems.

1 Introduction

The aim of this paper is to discuss the effectiveness of some iterative algorithms for solving the symmetric indefinite system that arises when we solve by an Interior–Point (IP) approach a large scale nonlinear programming (NLP) problem, of the form

$$\begin{array}{l} \min f(\boldsymbol{x}) \\ \boldsymbol{g}_1(\boldsymbol{x}) = 0 \\ \boldsymbol{g}_2(\boldsymbol{x}) \geq 0, \end{array} \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$, $f(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$, $\boldsymbol{g}_1(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^{neq}$, $\boldsymbol{g}_2(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}^m$ are twice continuously differentiable and the first and second derivatives of the objective function and constraints are available.
The idea of IP methods is based on the introduction of a slack vector $\boldsymbol{s} \in \mathbb{R}^m$ and on the transformation of the original problem in to a sequence of problems

[*]This research was supported by the Italian Ministry for Education, University and Research (MIUR), FIRB Project RBAU01JYPN.

with logarithmic barrier function, depending of a positive penalty parameter ρ that asymptotically goes to 0:

$$\begin{array}{l} \min f(\boldsymbol{x}) - \rho_k \sum_{k=1}^{m} \ln s_i \\ \boldsymbol{g}_1(\boldsymbol{x}) = 0 \\ \boldsymbol{g}_2(\boldsymbol{x}) - \boldsymbol{s} = 0. \end{array} \tag{2}$$

The basic step of an IP scheme is to determine by one Newton–type iteration an approximate solution of the nonlinear system that gives the Karush–Kuhn–Tucker (KKT) optimality conditions of the problem (2)

$$\left(\begin{array}{l} \nabla f(\boldsymbol{x}) - \nabla \boldsymbol{g}_1(\boldsymbol{x})\boldsymbol{\lambda}_1 - \nabla \boldsymbol{g}_2(\boldsymbol{x})\boldsymbol{\lambda}_2 \\ -\boldsymbol{g}_1(\boldsymbol{x}) \\ -\boldsymbol{g}_2(\boldsymbol{x}) + \boldsymbol{s} \\ \Lambda_2 S \boldsymbol{e}_m \end{array} \right) = \rho \tilde{\boldsymbol{e}}. \tag{3}$$

with

$$\boldsymbol{s} \geq 0 \qquad \boldsymbol{\lambda}_2 \geq 0,$$

or, in a more concise notation,

$$\boldsymbol{H}(\boldsymbol{v}) = \rho \tilde{\boldsymbol{e}}$$

$$\boldsymbol{s} \geq 0 \qquad \boldsymbol{\lambda}_2 \geq 0,$$

where $\boldsymbol{\lambda}_1 \in \mathbb{R}^{neq}$ and $\boldsymbol{\lambda}_2 \in \mathbb{R}^m$ are vectors of Lagrange multipliers, $\Lambda_2 = \mathrm{diag}(\boldsymbol{\lambda}_2)$, $S = \mathrm{diag}(\boldsymbol{s})$, $\boldsymbol{v} = (\boldsymbol{x}^T, \boldsymbol{\lambda}_1^T, \boldsymbol{\lambda}_2^T, \boldsymbol{s}^T)$, $\boldsymbol{e}_m$ indicates the vector of m components whose values are equal to 1 and $\tilde{\boldsymbol{e}} = (\boldsymbol{0}_{n+neq+m}^T, \boldsymbol{e}_m^T)^T$. For a detailed explanation of an IP scheme see [11], [26], [22, Section 14].
The more time–consuming task of the k–th iteration of an IP method consists in applying a step of the Newton algorithm to system (3), determining the numerical solution of the following Newton linear equation

$$H'(\boldsymbol{v}^{(k)})\Delta \boldsymbol{v} = -H(\boldsymbol{v}^{(k)}) + \rho_k \tilde{\boldsymbol{e}}, \tag{4}$$

where, omitting the index iteration k, the jacobian matrix of $\boldsymbol{H}(\boldsymbol{v})$ is given by

$$H'(\boldsymbol{v}) = \left(\begin{array}{cccc} Q & B & C & 0 \\ B^T & 0 & 0 & 0 \\ C^T & 0 & 0 & I \\ 0 & 0 & S & \Lambda_2 \end{array} \right), \tag{5}$$

with $Q = \nabla^2 f(\boldsymbol{x}) - \sum_1^{neq} \lambda_{1,i} \nabla^2 g_{1,i}(\boldsymbol{x}) - \sum_1^m \lambda_{2,i} \nabla^2 g_{2,i}(\boldsymbol{x})$, $B = -\nabla \boldsymbol{g}_1(\boldsymbol{x})$ and $C = -\nabla \boldsymbol{g}_2(\boldsymbol{x})$. Here Q is the Hessian matrix of the Lagrangian function of the problem (2), $\nabla^2 f(\boldsymbol{x})$, $\nabla^2 g_{1,i}(\boldsymbol{x})$, $\nabla^2 g_{2,i}(\boldsymbol{x})$ are the Hessian matrices of the function $f(\boldsymbol{x})$ and of the i–th component of the constraints $\boldsymbol{g}_1(\boldsymbol{x})$, and $\boldsymbol{g}_2(\boldsymbol{x})$ respectively; then, $\lambda_{1,i}$ and $\lambda_{2,i}$ are the i–th component of $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$ respectively.

Let assume $H'(\boldsymbol{v})$ be a nonsingular matrix. The strategy used in the IP method updates the iterate by a convenient damping parameter which guarantees that $\boldsymbol{\lambda}_2$ and $\boldsymbol{s}$ are preserved strictly positive at any iteration.
From the last block of equations of (4), we can deduce

$$\Delta \boldsymbol{s} = \Lambda_2^{-1}[-S\Delta\boldsymbol{\lambda}_2 - \boldsymbol{\theta} + \rho \boldsymbol{e}_m],$$

where $\boldsymbol{\theta} = \Lambda_2 S\boldsymbol{e}_m$ and, then, the system (4) can be rewritten in *reduced form*

$$\begin{pmatrix} Q & B & C \\ B^T & 0 & 0 \\ C^T & 0 & -\Lambda_2^{-1}S \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\lambda}_1 \\ \Delta\boldsymbol{\lambda}_2 \end{pmatrix} = \begin{pmatrix} -\boldsymbol{\alpha} \\ -\boldsymbol{\beta} \\ \boldsymbol{g}_2(x) - \rho\Lambda_2^{-1}\boldsymbol{e}_m \end{pmatrix}, \qquad (6)$$

with $\boldsymbol{\alpha} = \nabla f(\boldsymbol{x}) - \nabla \boldsymbol{g}_1(\boldsymbol{x})\boldsymbol{\lambda}_1 - \nabla \boldsymbol{g}_2(\boldsymbol{x})\boldsymbol{\lambda}_2$ and $\boldsymbol{\beta} = -\boldsymbol{g}_1(\boldsymbol{x})$.
By a further substitution from the third block equation, we have

$$\Delta\boldsymbol{\lambda}_2 = S^{-1}[\Lambda_2 C^T\Delta\boldsymbol{x} + \Lambda_2\boldsymbol{g}_2(\boldsymbol{x}) + \rho\boldsymbol{e}_m].$$

Then, the system can be written in *condensed form*

$$\begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\lambda}_1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{q} \end{pmatrix}, \qquad (7)$$

and

$$\begin{aligned} A &= Q + CS^{-1}\Lambda_2 C^T \\ \boldsymbol{c} &= -\boldsymbol{\alpha} - CS^{-1}[-\Lambda_2\boldsymbol{g}_2(\boldsymbol{x}) + \rho\boldsymbol{e}_m] \\ \boldsymbol{q} &= -\boldsymbol{\beta}. \end{aligned}$$

Both the systems (6) or (7) are symmetric and indefinite and they can be solved by the sparse Bunch–Parlett triangular factorization ([4]), that combines dynamic reordering for sparsity preserving and pivoting technique for numerical stability (see routine MA27 of HSL Library ([10])).
Nevertheless, for large scale NLP problems, the size of these systems is large and, even if the coefficient matrices are sparse and the sparsity is exploited, the computation of the exact solution by direct methods can be very expensive in terms of CPU time and storage requirements. In Table 1 we report the numerical results, in terms of number of iterations (it.) and execution time (in seconds), of the IP method that uses the routine MA27 for solving (7), obtained on a subset of test–problems described in Table 2. Only the test–problems of smallest size (n+neq up to 100000) can be solved, but the execution time very quickly increases. For larger test–problems (not reported in Table 1), we observed a failure after a few iterates, due to fill–in of the factor which exceeds the available memory. Indeed, the Gauss factor computed by the routine MA27 does not depend only on the matrix structure and at each iteration the fill–in can change.
For the above reasons, in the framework of direct methods, much efforts have been performed to avoid the use of MA27 for large scale NLP problems. Some

IP-MA27

Prob.	iter	time	Prob.	iter.	time
TPB1-99	29	27.38	TPB6-99	24	23.1
TPB1-199	37	349.66	TPB6-199	26	258.7
TPB2-99			TPB7-99	26	22.1
TPB2-199	35	339.1	TPB7-199	31	269.1
TPB3-99	24	22.52	TPB8-99	27	22.9
TPB3-199	27	250.28	TPB8-199	33	285.1
TPB4-99	25	22.7	TPB9-119	31	48.1
TPB4-199	30	269.7	TPB9-179	34	406.7
TPB5-99	24	21.7	TPB10-119	35	54.6
TPB5-199	26	370	TPB10-179	40	581.5
TPD6- 99	25	24.71			
TPD6-199	26	304.11			

Table 1: Control problems with direct inner solver

IP solvers transform the symmetric systems (6) or (7) into a *quasidefinite* form[1] ([25]), so that a Cholesky–like factorization can be obtained. At the start of the IP scheme, the a–priori determination of a sparsity preserving reordering of the coefficient matrix (taking into account only of its structure) and of the symbolic Cholesky factor is carried out. Then, at each iteration the factor is computed, without using pivoting technique, saving a lot of CPU time. The reduction of a coefficient matrix into a quasidefinite form is obtained by a *regularization* technique, consisting in to perturb this matrix by adding a convenient diagonal matrix R. Different ways to construct R are proposed: see, for example, [26], [24], or [1]. In this last paper, R is dynamically computed by a very simple procedure, that can be easily included in the implementation of the Cholesky factorization: when a critical pivot is reached, this is perturbed by a small quantity with a convenient sign.
Nevertheless, the use of regularization requires additional recovery procedures and several factorizations (for example to individuate a perturbation as small as possible ([26]) or to implement an iterative refinement if the computed solution of the perturbed system is not satisfactory ([1]), etc.).
A different approach that avoids modifications of the matrices of the subproblems is to use iterative inner solvers for (6) or (7), that exploit the sparsity of the involved matrices, solving *approximately* the inner subproblems, so that unnecessary inner iterations can be avoided when we are far from the solution. In some recent papers, the IP scheme combined by an *inexact* inner solver can

[1]A matrix $\begin{pmatrix} S & V \\ V^T & -U \end{pmatrix}$ is quasidefinite if S and U are symmetric positive definite matrices. A quasidefinite matrix is strongly factorizable, i. e. a Cholesky–like factorization LDL^T (with a diagonal matrix D and a lower triangular matrix L with diagonal elements equal to one) exists for any symmetric permutation of the quasidefinite matrix. The diagonal matrix D has a number of positive (negative) diagonal entries equal to the size of S (U respectively).

be viewed as an Inexact Newton method scheme ([9], [7], [3]). From this interpretation, it is possible to deduce a suitable *adaptive* stopping rule for the inner solver that assures the global convergence and the local superlinear convergence of the whole outer–inner scheme.
In this paper we discuss about the effectiveness of two iterative methods for the solution of symmetric indefinite systems, that allow an *a priori* symbolic factorization avoiding the pivoting technique needed in the MA27 subroutine. In particular, in Section 2, we consider the iterative Hestenes' multipliers scheme. This algorithm leads the solution of the system (7) to that of a sequence of smaller symmetric positive definite systems, so efficient sparse Cholesky codes can be used.
In Section 3, we propose two different implementations of the preconditioned conjugate gradient (PCG) algorithm for (7) with the preconditioner described in [16] (see also [17]). The solution of the systems related to the preconditioner is performed by a sparse Cholesky factorization in the first case and by a sparse Cholesky–like factorization in the second version. This last version does not require the computation of matrix–matrix products as in the first version and in the Hestenes' multipliers scheme. By using a regularization technique, we dynamically compute a preconditioner that admits a Cholesky–like factorization, maintaining the well known features of the efficient sparse Cholesky codes.
In the Section 4, numerical results obtained by a code implementing the IP method combined with Hestenes' multipliers scheme or PCG algorithm, are given for a selection of very large test–problems, arising from the discretization of semielliptic control problems in [19], [20], [21]. In this case, we deal with NLP problems with equality and simple box constraints, with very sparse and structured matrices in (5). The IP method combined with the PCG algorithm that uses the second version of the preconditioner (IP–PCG2) enables us to efficiently solve semielliptic control problems with size n+neq up to 700000.

2 The Hestenes' multipliers scheme for the solution of the condensed KKT system

When we have to solve NLP problems as those in [19], [20], [21], where *the inequality constraints are simple box constraints*, it is convenient to reduce the inner linear system (4) in the form (7); indeed, in this case, the term $C^T S^{-1}\Lambda_2 C$ of the matrix A is easily computable since it is a diagonal matrix.
It is well known that, if B^T is a full row–rank matrix, the coefficient matrix of (7)

$$M = \begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix}$$

is nonsingular if and only if the matrix A is nonsingular on the null space of B^T ([12]), i.e. $Z^T AZ$ is a nonsingular matrix, where Z is the $n \times (n - neq)$ matrix such that $B^T Z = 0$ and $Z^T Z = I$. In particular, a sufficient condition for the nonsingularity of M is that the matrix $Z^T AZ$ is positive definite (see also [15,

p. 424]). This condition holds if the hessian matrix of the lagrangian function of the problem (1) is positive definite on the null space of B^T. Note that this assumption is also the one required for the local SQP method ([22, p. 531]).
Setting $\boldsymbol{y}_1 = \Delta\boldsymbol{x}$ and $\boldsymbol{y}_2 = \Delta\boldsymbol{\lambda}_1$, the system (7), can be viewed as the Lagrange necessary conditions for the minimum point of the following quadratic problem

$$\begin{array}{ll} min & \frac{1}{2}\boldsymbol{y}_1^T A\boldsymbol{y}_1 - \boldsymbol{c}^T\boldsymbol{y}_1 \\ & B^T\boldsymbol{y}_1 - \boldsymbol{q} = 0. \end{array}$$

This quadratic problem can be solved efficiently by Hestenes' multipliers scheme ([13, p. 308]), that consists in updating the dual variable by the rule

$$\boldsymbol{y}_2^{(j+1)} = \boldsymbol{y}_2^{(j)} + \chi(B^T\boldsymbol{y}_1^{(j)} - \boldsymbol{q}),$$

where χ is a positive parameter (penalty parameter) and $\boldsymbol{y}_1^{(j)}$ minimizes the augmented lagrangian function of the quadratic problem

$$\mathcal{L}_\chi(\boldsymbol{y}_1, \boldsymbol{y}_2) = \frac{1}{2}\boldsymbol{y}_1^T A\boldsymbol{y}_1 - \boldsymbol{y}_1^T\boldsymbol{c} + \boldsymbol{y}_2^T(B^T\boldsymbol{y}_1 - \boldsymbol{q}) + \frac{\chi}{2}(B^T\boldsymbol{y}_1 - \boldsymbol{q})^T(B^T\boldsymbol{y}_1 - \boldsymbol{q}).$$

This means that $\boldsymbol{y}_1^{(j)}$ is the solution of the linear system of order n

$$(A + \chi BB^T)\boldsymbol{y}_1 = -B\boldsymbol{y}_2^{(j)} + \boldsymbol{c} + \chi B\boldsymbol{q} \tag{8}$$

Note that, since B^T has full row–rank, the null space of BB^T is equal to the null space of B^T; then the matrix A is positive definite on the null space of BB^T. Then, it is immediate the following theorem.

Theorem 2.1 ([15, p. 408]) There exists a positive parameter χ^* such that for all $\chi > \chi^*$, the matrix $A + \chi BB^T$ is positive definite.

This result enables us to solve the system (8) by applying a Cholesky factorization.
In order to choose the parameter χ, we observe that, for any $\boldsymbol{x} \neq 0$, we must have $\boldsymbol{x}^T(A + \chi BB^T)\boldsymbol{x} > 0$. When $B^T\boldsymbol{x} = 0$, we have $\boldsymbol{x}^T A\boldsymbol{x} > 0$. If $B^T\boldsymbol{x} \neq 0$, $\boldsymbol{x}^T BB^T\boldsymbol{x} > 0$. Then, it follows that

$$\chi > \max(0, \max_{\boldsymbol{x}\notin\mathcal{N}(B^T)} \frac{-\boldsymbol{x}^T A\boldsymbol{x}}{\boldsymbol{x}^T BB^T\boldsymbol{x}})$$

Since $\|A\| \geq (-\boldsymbol{x}^T A\boldsymbol{x})/(\boldsymbol{x}^T\boldsymbol{x})$ for any natural norm and also for the Frobenius norm $\|\cdot\|_F$, and $\boldsymbol{x}^T BB^T\boldsymbol{x}/(\boldsymbol{x}^T\boldsymbol{x}) \geq \tau_{min}$, where τ_{min} is the minimum nonzero eigenvalue of BB^T or of $B^T B$, we can choose as χ the following value:

$$\chi > \frac{\|A\|_F}{\tau_{min}}$$

In general it is difficult to determine an estimate of τ_{min}. Numerical evidence shows that a good approximation of τ_{min} is $\min(1, t_{min})$, where t_{min} is the minimum diagonal entry of the matrix $B^T B$, although $t_{min} \geq \tau_{min}$. Furthermore, in order to avoid that the value of χ is too small (the matrix is not positive definite) or too large (too ill–conditioned system), it is convenient to use safeguards. In the numerical experiments of the last section, the following value of χ produced good results:

$$\chi = \min(\max(10^7, \frac{\max\{\|A\|_F, 1\}}{\min\{t_{min}, 1\}}), 10^8). \tag{9}$$

Now, we discuss the implementation of the method. We assume that the hessian matrix Q of the lagrangian function and the jacobian matrix B^T of the equality constraints are stored in a column compressed format ([23]). The matrices A and Q have the same structure and are different only for the diagonal entries, since we assume that the inequality constraints are box constraints and, consequently, $CS^{-1}\Lambda_2 C^T$ is a diagonal matrix.
Then, at any step of the IP method, the implementation of Hestenes' multipliers scheme requires the computation of the matrix $T = A + \chi BB^T$ and its Cholesky factorization $T = L_n L_n^T$. The other operations related to each iteration (i. e. sparse matrix–vector products $B(-\boldsymbol{y}_2^{(j)} + \chi \boldsymbol{q})$ and $B^T \boldsymbol{y}_1^{(j)}$ and solution of the triangular systems equivalent to (8)) have a negligible computational complexity. In order to execute only necessary operations to form T, it is convenient to execute a preprocess procedure that builds a data structure which stores the indices of the nonzero entries of the lower triangular part of the symmetric matrix T. For any nonzero entry t_{ij}, $i \leq j$ of T, in the same data structure we also store the pairs of indices of the elements of B and B^T that give a nonzero contribution in the scalar product forming the entry, as depicted in Figure 1. The preprocess routine also computes the symbolic Cholesky factorization of the sparse, symmetric and positive definite matrix T. To exploit the sparsity of T, its factorization can be obtained by a very efficient Fortran package (version 0.3) of Ng and Peyton (included in the package LIPSOL, downloadable from *www.caam.rice.edu/˜zhang/lipsol*). This package *a priori* computes the symbolic factor of T (i.e. the indices of the nonzero entries of L_n and the information to form these entries), using the multiple minimum degree ordering of Liu to minimize the fill–ins in L_n and the supernodal block factorization to take advantage of the presence of the cache memory in modern computer architectures ([14]). The *a priori* procedure of Liu for the reordering of T and the computation of its symbolic factorization is executed only one time in the preprocess routine.
In conclusion, the time for solving an NLP problem by the IP method combined with Hestenes' multipliers method is subdivided in two part, the *preprocess time*, that is the time needed to determine the data structure of the nonzero entries of T and to compute the symbolic Cholesky factorization of T, and the time for computing the solution (*solution time*). We observe that the preprocess time is dependent on the strategy used to perform the matrix–matrix products needed

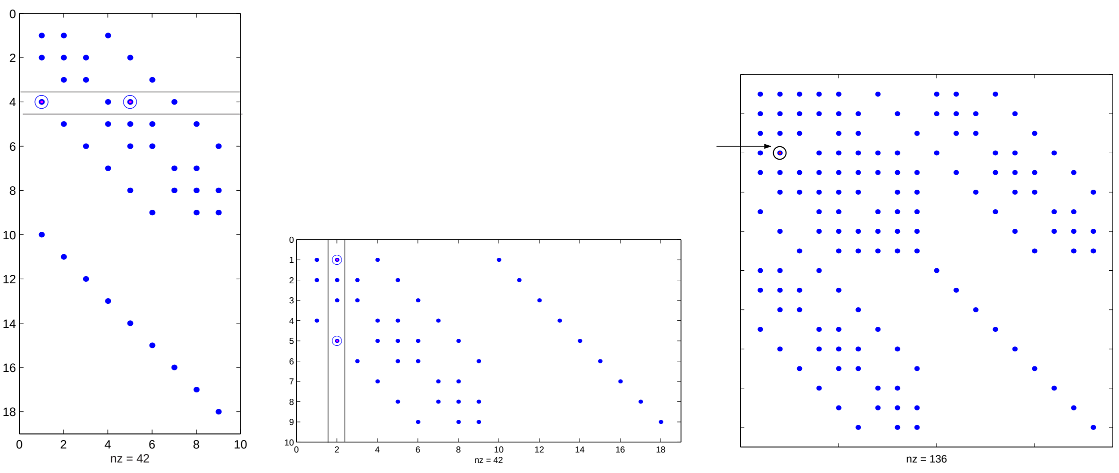

Figure 1: Preprocess phase: save the indices of the nonzero contribution of the scalar product for determining $t_{4,2}$ (and $t_{2,4}$) in the matrix–matrix product BB^T. We save the couples of indices of the vectors containing the nonzero entries of the matrices B and B^T related to the elements denoted by a circle.

in the method for computing T. Following our approach, the time needed for building the data structure of indices described above and in Figure 1 is the 99% of the whole preprocess time. Then, exploiting the data structure, the matrix–matrix product performed at each iteration has a cheap computational cost, at most the 15% (the 5% for the larger problem sizes) of the whole solution time.

3 The Preconditioned Conjugate Gradient method for the solution of the KKT system

A different approach for solving the inner system arising at each step of an IP scheme uses a Preconditioned Conjugate Gradient (PCG) method, as suggested in [16] (see also [9], [8], [17], [2], [6]). As in the previous section, we propose to solve the condensed form of the system (7) instead of the reduced form (6), but, unlike as it arises for the Hestenes' multipliers scheme, in this case we can avoid to explicitly compute the matrix $A = Q + CS^{-1}\Lambda_2 C^T$. Indeed, at any step of the PCG scheme, the matrix A is required only in the matrix–vector product $t = Mp$, where

$$M = \begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix}, \qquad p = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}, \qquad p_1 \in \mathbb{R}^n, \quad p_2 \in \mathbb{R}^{neq}.$$

The product Mp can be executed by sparse matrix–vector products only, using a temporary array $\hat{\boldsymbol{t}}$ to store the partial results:

$$\begin{array}{rcl} \boldsymbol{t}_1 & \leftarrow & C^T\boldsymbol{p}_1 \\ \hat{\boldsymbol{t}} & \leftarrow & S^{-1}\Lambda_2\boldsymbol{t}_1 \\ \boldsymbol{t}_1 & \leftarrow & C\hat{\boldsymbol{t}} \\ \boldsymbol{t}_1 & \leftarrow & \boldsymbol{t}_1 + Q\boldsymbol{p}_1 + B\boldsymbol{p}_2 \\ \boldsymbol{t}_2 & \leftarrow & B^T\boldsymbol{p}_1 \end{array}$$

As preconditioner in the PCG scheme, we can consider the indefinite preconditioner in [16]:

$$\bar{M} = \begin{pmatrix} \bar{A} & B \\ B^T & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ B^T\bar{A}^{-1} & I \end{pmatrix} \begin{pmatrix} \bar{A} & 0 \\ 0 & -B^T\bar{A}^{-1}B \end{pmatrix} \begin{pmatrix} I & \bar{A}^{-1}B \\ 0 & I \end{pmatrix} \tag{10}$$

where we assume that $\bar{A}$ is a positive diagonal approximation of A. For sake of completeness, we report the main theoretical results about the preconditioner (10)(for further details and proofs of the following theorems, see [16]).

Theorem 3.1 If $\bar{A}$ is a positive definite matrix , then the matrix $M\bar{M}^{-1}$ has at least $2 \cdot neq$ unit eigenvalues. If $A\bar{A}^{-1} - I$ is a nonsingular matrix, then only neq linearly independent eigenvectors corresponding to these eigenvalues exist; the other eigenvalues of the matrix $M\bar{M}^{-1}$ are exactly the eigenvalues of the matrix $Z^TAZ(Z^T\bar{A}Z)^{-1}$. If Z^TAZ is a positive definite matrix, all the eigenvalues of the matrix $M\bar{M}^{-1}$ are positive. Moreover, if $\boldsymbol{v}Z^T\bar{A}Z\boldsymbol{v} = \boldsymbol{v}^TZ^TAZ\boldsymbol{v}$ for some $\boldsymbol{v} \in \mathbb{R}^n$, then all the eigenvalues of the matrix $M\bar{M}^{-1}$ are included in the interval determined by the extremal eigenvalues of the matrix $Z^TAZ(Z^T\bar{A}Z)^{-1}$.

Theorem 3.2 Consider the PCG method with preconditioner (10), where the matrix $\bar{A}$ is positive definite, applied to the system

$$M \begin{pmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{pmatrix}.$$

If a breakdown does not occur, then we obtain the solution $\begin{pmatrix} \boldsymbol{v}_1^* \\ \boldsymbol{v}_2^* \end{pmatrix}$ after at most $n - neq + 2$ iterations.

Theorem 3.3 Let the matrix Z^TAZ be positive definite. Consider the PCG method with the preconditioner (10), where $\bar{A}$ is a positive definite matrix, applied to the system (7), starting with the initial point $v_1^0 = \bar{A}^{-1}B(B^T\bar{A}^{-1}B)^{-1}y_2$, $v_2^0 = 0$. The PCG method finds the solution of the system after at most $n - neq$ iterations and the following condition holds

$$\|\boldsymbol{v}_1^i - \boldsymbol{v}_1^*\| \le 2\sqrt{k}\left(\frac{1-\sqrt{k}}{1+\sqrt{k}}\right)^i \|\boldsymbol{v}_1^0 - \boldsymbol{v}_1^*\| \tag{11}$$

where k is the spectral condition number of $Z^T AZ(Z^T \bar{A}Z)^{-1}$.

In the implementation of the PCG scheme, we can choose the diagonal matrix $\bar{A} = diag(\bar{a}_{ii})$ as follows

$$\bar{a}_{ii} = \begin{cases} a_{ii} = q_{ii} + \sum_{k=1}^{m} c_{ik}^2 \lambda_{2,k}/s_k & \text{if } a_{ii} > 10^{-8} \\ 1.5 \cdot 10^{-8} & \text{otherwise.} \end{cases} \quad i = 1, ..., n \tag{12}$$

At any step of the PCG scheme, we have to compute the solution of the system

$$\bar{M} \begin{pmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_1 \\ \boldsymbol{r}_2 \end{pmatrix}. \tag{13}$$

We can determine the solution of this system in two different ways that produce a very different performance, especially for large scale problems.
In the first case (IP-PCG1), at the beginning of the PCG method we compute the symmetric positive definite matrix $T = B^T \bar{A}^{-1} B$ and its Cholesky factorization $T = L_{neq} L_{neq}^T$; then, computing $\bar{M}^{-1}$ by means of (10), the solution of (13) can be determined by the following procedure

$$\begin{array}{lcl} \boldsymbol{z}_1 & \leftarrow & \bar{A}^{-1}\boldsymbol{r}_1 \\ \boldsymbol{z}_2 & \leftarrow & \boldsymbol{r}_2 - B^T \boldsymbol{z}_1 \\ \boldsymbol{t}_2 & \leftarrow & -L_{neq}^{-1}\boldsymbol{z}_2 \\ \boldsymbol{z}_2 & \leftarrow & L_{neq}^{-T}\boldsymbol{t}_2 \\ \boldsymbol{z}_1 & \leftarrow & \boldsymbol{z}_1 - \bar{A}^{-1}B\boldsymbol{z}_2 \end{array}$$

where $\boldsymbol{t}_2$ is an neq–vector used to store the partial products.
As in the implementation of Hestenes' method, a preprocess routine can build a data structure that stores the information needed to compute the nonzero contribution to each nonzero scalar product. The preprocess routine can also determine the minimum degree reordering of the matrix T and its symbolic Cholesky factor. For these last tasks and for computing the elements of L_{neq}, we can use the package of Ng and Peyton. With this approach, the preprocess phase is generally less expensive than that of the IP method combined with the Hestenes' multipliers scheme, even for NLP problems with equality and box constraints. Indeed, we have to compute the entries of the matrix T and to solve systems with T as coefficient matrix, whose size is neq instead of the size n of the matrix $A + \chi BB^T$, where $neq < n$. Also in this case, the time to determine the data structure for the indices of the nonzero entries of T is the 99% of the whole preprocess time.

Now, we discuss the other way to implement the PCG algorithm that avoids the computation of the matrix–matrix product $B^T \bar{A}^{-1} B$. We call this second version of the PCG algorithm IP-PCG2.
We observe that the matrix $\bar{M}$ can be factorized in a Cholesky–like form

$$L_{n+neq} D L_{n+neq}^T, \tag{14}$$

where L_{n+neq} is a lower triangular matrix with diagonal entries equal to one and D is a nonsingular diagonal matrix. In order to reduce the fill–ins in the lower triangular factor, we can perform a minimum degree reordering of the matrix $\bar{M}$. But, it is not assured that the symmetrically permuted matrix $P\bar{M}P^T$ can be factorized in the Cholesky–like form.
Nevertheless, we can obtain a factorization in the form (14) if we use for the matrix $\bar{M}$ the regularization technique described in [1]; in other words, instead of using the preconditioner $\bar{M}$, we compute the factorization of

$$\bar{\bar{M}} = \bar{M} + \begin{pmatrix} R_1 & 0 \\ 0 & -R_2 \end{pmatrix}$$

where R_1 and R_2 are non negative diagonal matrices such that $P\bar{\bar{M}}P^T$ admits a factorization of the form (14). The computation of R_1 and R_2 can be obtained during the factorization procedure. If a pivot d_i is too small ($|d_i| < 10^{-15} \max_{j<i} |d_j|$), we put $d_i = \sqrt{\epsilon}$ if $1 \leq i \leq n$, or $d_i = -\sqrt{\epsilon}$ if $n+1 \leq i \leq n+neq$, where ϵ is the machine precision.
The dynamic computation of the elements of R_1 and R_2 reduces the perturbation to a minimum. This approach is used in [2] for linear and quadratic programming problems with equality and box constraints.
The Cholesky–like factorization of $\bar{\bar{M}}$ can be obtained by a modification of the Ng and Peyton package. In particular, we modify the subroutine PCHOL so that we compute $L_{n+neq}DL^T_{n+neq}$ with diagonal elements of L_{n+neq} equal to 1. Consequently, it is necessary to construct suitable subroutines (MMPYM and SMXPYM) to update the blocks of the factor L_{n+neq}, and to modify the subroutine BLKSVT for the computation of the solution of the system

$$L_{n+neq}DL^T_{n+neq}\boldsymbol{z} = \boldsymbol{r}.$$

The routines for performing the minimum degree reordering, for determining the supernodes and for the computation of the symbolic factor are unchanged. Consequently, the effectiveness of the package of Ng and Peyton due to a suitable use of the cache memory is maintained. This new package, called BLKFCLT, is downloadable from *dm.unife.it/blkfclt.*

4 Numerical Results

In order to evaluate the effectiveness of the Hestenes' multipliers scheme and the two versions of the PCG method, a Fortran 90 code, implementing the IP method described in [3] with different inner solvers, has been carried out on HP zx6000 workstation with Itanium2 processor 1.3 GHz and 2 Gb of RAM. The code has been compiled with a +O3 optimization option of the Fortran HP compiler.
In this code, the hessian matrix Q of the lagrangian function and the jacobian matrices B^T and C^T of the equality and inequality constraints are stored in a column compressed format ([23]).

The Newton IP method stops when

$$\|\boldsymbol{H}(\boldsymbol{v}^{(k)})\| \leq 10^{-8},$$

or when (see [26])

$$\frac{|\text{gap}|}{1+|\text{gap}|} \leq 10^{-8},$$

where "gap" is the difference between the primal function $f(\boldsymbol{x})$ and the dual function

$$\begin{aligned} d(\boldsymbol{x},\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2) &= f(\boldsymbol{x}) - \boldsymbol{\lambda}_2^T\boldsymbol{g}_2(\boldsymbol{x}) - \boldsymbol{\lambda}_1^T\boldsymbol{g}_1(\boldsymbol{x}) - \nabla f(\boldsymbol{x})^T\boldsymbol{x} + \\ & + \begin{pmatrix} \boldsymbol{\lambda}_1^T & \boldsymbol{\lambda}_2^T \end{pmatrix} \begin{pmatrix} \nabla \boldsymbol{g}_1(\boldsymbol{x})^T \\ \nabla \boldsymbol{g}_2(\boldsymbol{x})^T \end{pmatrix} \boldsymbol{x}. \end{aligned}$$

The inner solvers stop if the residual of the system (7) at the k–th iteration is such that

$$\|\boldsymbol{r}^{(k)}\| \leq \max(5\cdot 10^{-8}, \delta_k\|\boldsymbol{H}(\boldsymbol{v}^{(k)})\|),$$

or if a maximum number is reached; for the Hestenes' multipliers scheme, the maximum number is fixed equal to 15, while for the PCG method, it is equal to the size of the system $n + neq$; for the value of δ_k see [3].
Numerical experiments have been carried out using the code on a set of semielliptic control problems described in [19], [20] and [21]. These problems by a suitable finite–difference discretization can be transcribed into large scale finite–dimensional NLP problems, where the objective function often is a quadratic form, the elliptic state equation and the Dirichlet and/or Neumann boundary conditions become equality constraints and the control and state constraints are simple box constraints. Then, in all test–problems, the matrix $CS^{-1}\Lambda_2C^T$ is a simple diagonal matrix whose computation is inexpensive for any inner solver.
In Table 2, we report the references of the considered test–problems. The 'B' symbol in 'TPB*-N' indicates that the problem has a boundary control, while the distributed ones are indicated with the letter 'D'.
The number of variables n and the number of the equality constraints neq depend on a parameter N which represents the number of the mesh points for each dimension of the square domain of the control problem. The suffix in the name of the test–problems is the value of N.
In Tables 3 and 4, for each test–problems, we report the values of n, neq, the number of lower (nl) and upper (nu) bounds and the number of nonzero entries $nnzq$ and $nnzb$ of Q and B respectively. Then, in Table 5 we have:

- for the Hestenes' scheme (IP-Hestenes) the number $nnzhes$ of the nonzero entries of the lower triangular part of $A + \chi BB^T$ and the number $Lhes$ of the nonzero entries of its Cholesky factor;
- for the first version of the PCG method (IP-PCG1) the number $nnzpcg1$ of the nonzero entries of the lower triangular part of $B^T\bar{A}^{-1}B$ and the number $Lpcg1$ of the nonzero entries of its Cholesky factor;

- for the second version of the PCG method (IP-PCG2) the number $nnzpcg2$ of the nonzero entries of the lower triangular part of $\bar{M}$ and the number $Lpcg2$ of the nonzero entries of D and of the strictly lower part of the Cholesky–like factor L_{n+neq}.

We observe that, in IP-Hestenes, because of the structure of B, the matrix–matrix product BB^T does not give rise to an excessive number of nonzero entries and the matrix $A + \chi BB^T$ is very sparse with a density at most equal to 0.1%. Furthermore the ratio of the nonzero entries in the Cholesky factor and in the lower part of the matrix $A + \chi BB^T$ is at most equal to 15.3. The same considerations hold in IP-PCG1 for the matrix–matrix product $B^T\bar{A}^{-1}B$ and its Cholesky factor. Furthermore, the nonzero entries of $B^T\bar{A}^{-1}B$ and of its Cholesky factor are less than those of $A + \chi BB^T$ and of its Cholesky factor respectively. For the case IP-PCG2, the number of nonzero elements of the matrix D and of the Cholesky–like factor L_{n+neq} are not significantly different from those of the Cholesky factor of $\bar{M}$ for IP-PCG1.
In Tables 6, 7, 8, 9, 10 we report the results of the Newton IP method when we use as inner solvers the Hestenes multipliers' scheme (IP-Hestenes), the first version (IP-PCG1) and the second version (IP-PCG2) of the PCG method. In this table, *it* represents the number of outer iterations of Newton IP method. The total number of inner iterations of the inner solver is reported in brackets. For IP-Hestenes and IP-PCG1, the execution time, expressed in seconds, is subdivided into two parts, the *preprocess time* and the time for computing the solution (*solution time*). We recall that the preprocess routine performs the computation of the data structure employed at each iteration for the matrix-matrix product and the symbolic factorization of the matrix. The 99% of the preprocess time is spent in building the data structure for the matrix-matrix product. The results obtained show the effectiveness of the second version of the PCG solver (IP-PCG2), above all for very large–dimensional and sparse NLP problems. The code is efficient from the point of view of the memory usage and of the execution time. In the case of IP-Hestenes and of IP-PCG1, the more expensive computational task is the preprocess phase, which is dependent on the strategy used to perform the matrix–matrix products and on the size of the resulting matrices. Then, even if the IP-PCG2 code could perform more inner iterations than the IP-PCG1 version, the number of outer iterations is about equal in the two version of the IP method. Consequently the absence of the preprocess phase in the IP-PCG2 makes this method more efficient.
In some problems, when the meshsize is large, the number of outer iterations of the IP-Hestenes is large. A possible reason of this behaviour could be the ill conditioning of the matrix $A + \chi BB^t$. Indeed, at some iterations, the Hestenes inner solver cannot reach the required tolerance. In these cases, the inner iterations are anyway stopped after 15 steps, but the solution misses to satisfy the required tolerance (we observed that the residual is about 10 times grater). Obviously, in these situations, (TPB1,7,10, TPD3,6 for example) the direction provided by the Hestenes inner solver is not a "good" direction, and the algorithm "corrects" this mistake by performing more outer iterations. Otherwise,

we have a failure of the algorithm (see TPD1,2). In other cases (TPB8 for example), the situation is different, the Hestenes solver provides the solution satisfying the required tolerance but the number of outer iterations is grater than for the IP-PCG1 and IP-PCG2, and the previous explanation does not hold.
In the Table 11 we report some results, obtained by professor H. Mittelmann at the Arizona State University [18], of a comparison, in terms of execution time (in seconds) of the IP-PCG2 method with the version 3.1 of KNITRO-D (direct inner solver) and of KNITRO-I (iterative inner solver) [5] for solving the test problem TPB1. The numerical experiments have been carried out on a 3.2MHz Pentium 4 and the tolerance for KNITRO solvers has been set to 10^{-9}, in order to obtain the same precision on the final value of the objective function. Indeed, with these settings, the minimum computed by KNITRO coincides with the resulting value of IP-PCG2 on 8 significant figures, while with a tolerance of 10^{-8}, the value produced by KNITRO is greater than the resulting value of IP-PCG2. Table 11 shows that the better performances in terms of time are given by IP-PCG2 and for $N = 499$ KNITRO does not get the solution.

5 Conclusions

In the framework of the IP methods combined with inner iterative solvers, we devised a preconditioner $\bar{\bar{M}}$ for solving the system (7) by the PCG algorithm. The matrix $\bar{\bar{M}}$ is a dynamically computed regularized variant of the preconditioner $\bar{M}$ in [16], that does not require additional matrix–matrix products and that admits a Cholesky–like factorization (as a quasi–definite matrix), exploiting the well known techniques used to obtain an efficient implementation of the Cholesky algorithm (minimum degree reordering, determination of supernodes, use of cache memory). This Cholesky–like factorization can be computed by the routine BLKFCLT, downloadable from *dm.unife.it/blkfclt*. Following this approach, we were able to solve a set of semielliptic control problems with size n+neq up to 700000.

Acknowledgements The authors are extremely grateful to Prof. Hans Mittelmann for fruitful discussion and for the results of Table 11 and to the anonymous referee who stimulated us to improve the paper with his comments.

Table 2: Description of the test-problems.

Test problems	References
TPB1-N	[19], Example 5.5
TPB2-N	[19], Example 5.6
TPB3-N	[19], Example 5.7
TPB4-N	[19], Example 5.8
TPB5-N	[19], Example 5.1
TPB6-N	[19], Example 5.2
TPB7-N	[19], Example 5.3
TPB8-N	[19], Example 5.4
TPB9-N	[21], Example 4.1, $\alpha = 0.005$
TPB9-N	[21], Example 4.1, $\alpha = 0$
TPD1-N	[20], Example 1
TPD2-N	[20], Example 2
TPD3-N	[20], Example 3
TPD4-N	[20], Example 4
TPD5-N	[20], Example 5
TPD6-N	[21], 4.2, $M = 1, K = 0.8, b = 1, u_1 = 1.7, u_2 = 2, \psi(x) = 7.1$
TPD7-N	[21], 4.2, $M = 0, K = 1, b = 1, u_1 = 2, u_2 = 6, \psi(x) = 4.8$

Problem	n	neq	nu	nl	nnzb	nnzq
TPB1-99	10593	10197	10593	39204	50193	10593
TPB1-199	41193	40397	41193	158404	200393	41193
TPB1-299	91793	90597	91793	357604	450593	91793
TPB1-399	162393	160797	162393	636804	800793	162393
TPB1-499	252993	250997	252993	996004	1250993	252993
TPB1-599	363593	361197	363593	1435204	1801193	363593
TPB2-99	10593	10197	10593	39204	50193	10197
TPB2-199	41193	40397	41193	158404	200393	40397
TPB2-299	91793	90597	91793	357604	450593	90597
TPB2-399	162393	160797	162393	636804	800793	160797
TPB2-499	252993	250997	252993	996004	1250993	250997
TPB2-599	363593	361197	363593	1435204	1801193	361197
TPB3-99	10593	10197	10593	39204	50193	10593
TPB3-199	41193	40397	41193	158404	200393	41193
TPB3-299	91793	90597	91793	357604	450593	91793
TPB3-399	162393	160797	162393	636804	800793	162393
TPB3-499	252993	250997	252993	996004	1250993	252993
TPB3-599	363593	361197	363593	1435204	1801193	363593
TPB4-99	10593	10197	10593	39204	50193	9801
TPB4-199	41193	40397	41193	158404	200393	39601
TPB4-299	91793	90597	91793	357604	450593	89401
TPB4-399	162393	160797	162393	636804	800793	159201
TPB4-499	252993	250997	252993	996004	1250993	249001
TPB4-599	363593	361197	363593	1435204	1801193	358801
TPB5,6-99	10197	9801	10197	396	49005	10197
TPB5,6-199	40397	39601	40397	796	198005	40397
TPB5,6-299	90597	89401	90597	1196	447005	90597
TPB5,6-399	160797	159201	160797	1596	796005	160797
TPB5,6-499	250997	249001	250997	1996	1245005	250997
TPB5,6-599	361197	358801	361197	2396	1794005	361197
TPB7,8-99	10197	9801	10197	396	49005	9801
TPB7,8-199	40397	39601	40397	796	198005	39601
TPB7,8-299	90597	89401	90597	1196	447005	89401
TPB7,8-399	160797	159201	160797	1596	796005	159201
TPB7,8-499	250997	249001	250997	1996	1245005	249001
TPB7,8-599	361197	358801	361197	2396	1794005	358801
TPB9-119	14637	14518	14280	14637	71519	3840
TPB9-179	32757	32578	32220	32757	161279	8460
TPB9-279	78957	78678	78120	78957	390879	20160
TPB9-379	145157	144778	144020	145157	720479	36870
TPB9-479	231357	230878	229920	231357	1150079	58560
TPB9-579	337557	336978	335820	337557	1679679	85260
TPB10-119	14637	14518	14280	14637	71519	3721
TPB10-179	32757	32578	32220	32757	161279	8281
TPB10-279	78957	78678	78120	78957	390879	19881
TPB10-379	145157	144778	144020	145157	720479	36491
TPB10-479	231357	230878	229920	231357	1150079	58081
TPB10-579	337557	336978	335820	337557	1679679	84681

Table 3: Description of the test–problems: boundary control

Problem	n	neq	nu	nl	nnzb	nnzq
TPD1-99	19602	9801	19602	9801	59202	19602
TPD1-199	79202	39601	79202	39601	238402	79202
TPD1-299	178802	89401	178802	89401	537602	178802
TPD1-399	318402	159201	318402	159201	956802	318402
TPD1-499	498002	249001	498002	249001	1496002	498002
TPD2-99	19602	9801	19602	9801	59202	9801
TPD2-199	79202	39601	79202	39601	238402	39601
TPD2-299	178802	89401	178802	89401	537602	89401
TPD2-399	318402	159201	318402	159201	956802	159201
TPD2-499	498002	249001	498002	249001	1496002	249001
TPD3,4-99	19998	10197	19602	9801	59598	19602
TPD3,4-199	79998	40397	79202	39601	239198	79202
TPD3,4-299	179998	90597	178802	89401	538798	178802
TPD3,4-399	319998	160797	318402	159201	958398	318402
TPD3,4-499	499998	250997	498002	249001	1497998	498002
TPD5-99	19998	10197	19602	9801	59598	10197
TPD5-199	79998	40397	79202	39601	239198	40397
TPD5-299	179998	90597	178802	89401	538798	90597
TPD5-399	319998	160797	318402	159201	958398	160797
TPD5-499	499998	250997	498002	249001	1497998	250997
TPD6-99	19602	9801	19602	9801	58410	39204
TPD6-199	79202	39601	79202	39601	236810	158404
TPD6-299	178802	89401	178802	89401	535210	357604
TPD6-399	318402	159201	318402	159201	953610	636804
TPD6-499	498002	249001	498002	249001	1492010	996004
TPD7-99	19602	9801	19602	9801	58410	29403
TPD7-199	79202	39601	79202	39601	236810	118803
TPD7-299	178802	89401	178802	89401	535210	268203
TPD7-399	318402	159201	318402	159201	953610	477603
TPD7-499	498002	249001	498002	249001	1492010	747003

Table 4: Description of the test–problems: distributed control.

Problem Grid	nnzhes	Lhes	nnzpcg1	Lpcg1	nnzpcg2	Lpcg2
TPB1,2,3,4-99	70783	622759	69991	621571	60786	718637
TPB1,2,3,4-199	281583	3181444	279195	3179056	241586	3416032
TPB1,2,3,4-299	632383	8374469	628795	8370881	542386	9084296
TPB1,2,3,4-399	1123183	16252152	1118395	16247364	9631186	20102932
TPB1,2,3,4-499	1753983	26855490	1747995	26849502	1503986	28784753
TPB1,2,3,4-599	2524783	41135305	2517595	41128117	2164786	43488232
TPB5,6,7,8-99	69595	621571	67619	619595	59202	716261
TPB5,6,7,8-199	279195	3179056	275219	3175080	238401	3411256
TPB5,6,7,8-299	628795	8370881	622819	8364905	537602	9011520
TPB5,6,7,8-399	1118395	16247364	1110419	16239388	956802	20093356
TPB5,6,7,8-499	1747995	26849502	1738019	26839526	1496002	28772777
TPB5,6,7,8-599	2517595	41128117	2505619	41116141	2155202	43473654
TPB10,11-119	100315	945546	99720	944951	86156	1029560
TPB10,11-179	226075	2541572	225180	2540677	194036	2733190
TPB10,11-279	547675	7167732	546280	7166337	469836	8619291
TPB10,11-379	1009275	14501957	1007380	14500062	865636	15396152
TPB10,11-479	1610875	24901311	1608480	24898916	1381436	26203761
TPB10,11-579	2352475	37810473	2349580	37807578	2017236	48288922
TPD1,2,3-99	126029	715465	67619	619595	78012	735071
TPD1,2,3-199	512029	3409660	275219	3175080	316012	3488866
TPD1,2,3-299	1158029	8900195	622819	8364905	714012	9253530
TPD1,2,3-399	2064029	20090160	1110419	16239388	1272012	20405866
TPD1,2,3-499	3230029	28768781	1738019	26839526	1990012	29266787
TPD4,5-99	128401	717837	69595	621571	79596	737447
TPD4,5-199	516801	3414432	517993	3179056	319196	3493642
TPD4,5-299	1165201	8907367	1166993	8370881	718796	9260706
TPD4,5-399	2073601	20099732	2075994	16247364	1278396	20418142
TPD4,5-499	3242001	28780753	3244994	26849502	3244994	29278763
TPD6,7-99	72816	715465	67619	619595	78012	735071
TPD6,7-199	295620	3409660	510837	3175080	316012	3488866
TPD6,7-299	1158029	8900195	622819	8364905	714012	9079001
TPD6,7-399	2064029	20090160	1110419	16239388	1272012	20408566
TPD6,7-499	3230029	28768781	1738019	26839526	1990012	29266787

Table 5: Nonzero entries of the matrices and of the Cholesky factors

	IP-Hestenes			IP-PCG1			IP-PCG2	
Problem	it.(inn.)	time(prep.+iter.)	total	it.(inn.)	time(prep.+iter.)	total	it.(inn.)	total
TPB1-99	29(32)	2.2+2.0	4.3	37(30)	2.2+2.5	4.7	37(72)	5.2
TPB1-199	54(59)	36.4+22.9	59.3	45(37)	35.8+18.3	54.1	45(95)	38.9
TPB1-299	181(186)	206.4 +246.8	453.2	52(47)	197.3+68.1	256.4	52(116)	156.5
TPB1-399	327(341)	833.8+961.1	1794.9	58(53)	758.2+174.8	933.1	58(137)	493.1
TPB1-499	501(527)	1933.8+2768.7	4702.5	63(59)	1635.6+341.7	1977.4	63(158)	845.8
TPB1-599	*	*	*	66(62)	1902.2+701.2	2603.6	66(181)	1377.6
TPB2-99	34(34)	2.2+2.3	4.6	35(39)	2.3+2.4	4.7	35(37)	4.5
TPB2-199	40(40)	37.8+17.3	55.1	41(45)	37.8+17.3	55.1	41(41)	32.6
TPB2-299	55(56)	172.8+73.5	246.3	51(55)	201.8+68.13	269.9	50(52)	140.3
TPB2-399	143(144)	558.1+420.8	979	58(64)	655.9+171.6	827.5	59(60)	441.7
TPB2-499	197(198)	1418.3+1076.1	2494.4	66(76)	1621.4+364+9	1986.4	70(73)	829.1
TPB2-599	242(243)	2997.4+2367.5	5364.9	74(87)	3456.5+714.5	4171.1	79(89)	1560.2
TPB3-99	21(23)	3.02+1.5	4.5	29(36)	2.2+2.1	4.3	28(79)	4.3
TPB3-199	26(27)	47.8+10.9	58.7	33(42)	45.9+14.5	60.3	33(91)	30.0
TPB3-299	39(45)	162.2+52.9	215.0	36(47)	194.8+49.7	243.5	37(109)	115.8
TPB3-399	36(39)	831.0+105.3	936.3	39(54)	617.5+117.9	735.4	38(120)	312.8
TPB3-499	65(87)	2062.1+360.0	2422.2	42(55)	1522.1+232.9	1755.1	41(146)	535.1
TPB3-599	*	*	*	44(60)	3928.5+427.1	3928.5	43(159)	925.8
TPB4-99	31(31)	2.2+2.1	4.3	33(42)	2.3+2.3	4.6	31(44)	4.2
TPB4-199	38(98)	34.5+18.9	53.5	40(51)	36.6+17.2	53.8	38(59)	31.5
TPB4-299	41(58)	172.7+56.3	228.9	45(60)	189.9+61.3	251.3	40(59)	114.7
TPB4-399	43(45)	560.3+125.3	685.7	49(66)	603.6+147.2	750.8	45(67)	341.7
TPB4-499	*	*	*	51(67)	1499.3+283.7	1783.1	50(76)	601.7
TPB4-599	*	*	*	69(82)	3621.0+662.8	4284	82(153)	1660.4

Table 6: Numerical results: boundary control

	IP-Hestenes			IP-PCG1			IP-PCG2	
Problem	it.(inn.)	time(prep.+iter.)	total	it.(inn.)	time(prep.+iter.)	total	it.(inn.)	total
TPB5-99	28(28)	2.1+1.9	4	31(31)	2.1+2.1	4.2	28(34)	3.6
TPB5-199	33(33)	33.8+13.6	47.4	37(37)	35.7+15.2	50.9	32(42)	26
TPB5-299	40(55)	169.0+54.8	223.8	41(41)	194.9+53.9	248.8	36(50)	99.6
TPB5-399	45(75)	548.0+137.4	685.4	44(45)	751.0+128.1	879.1	39(62)	298.3
TPB5-499	49(79)	1380.6+275.1	1655.8	46(47)	1583.6+248.5	1832.1	43(73)	520.2
TPB5-599	51(126)	2978.9+534.4	3513.3	48(50)	3336.1+456.7	3792.9	46(77)	925.3
TPB6-99	30(30)	2.1+2.0	4.1	35(37)	2.1+2.4	4.5	30(39)	3.9
TPB6-199	33(33)	33.2+13.6	46.7	37(41)	35.4+15.4	50.8	32(41)	25.8
TPB6-299	40(55)	168.3+54.3	222.6	41(47)	231.6+55.8	287.4	37(54)	102.8
TPB6-399	46(61)	546.9+136.7	683.7	44(50)	634.8+129.4	764.2	40(65)	306.1
TPB6-499	50(95)	1377.3+288.2	1665.5	47(56)	1587.4+258.6	1846.1	44(75)	533.5
TPB6-599	51(126)	2971.5+532.8	3504.5	49(59)	3290.6+470.1	3760.8	46(77)	925.2
TPB7-99	39(39)	2.1+2.7	4.8	42(42)	2.1+2.8	4.9	40(54)	5.2
TPB7-199	46(46)	33.2+19.0	52.1	46(46)	35.6+18.9	54.5	45(72)	37.4
TPB7-299	64(99)	168.4+89.2	257.6	52(52)	193.6+69.5	263.1	49(87)	138.7
TPB7-399	96(141)	549.3+288.5	837.8	55(58)	636.0+160.5	796.5	53(95)	407.9
TPB7-499	127(169)	1387.3+703.4	2090.7	57(64)	1583.1+311.9	1895.1	52(94)	630
TPB7-599	159(202)	3020.9+1548.1	4569.1	59(69)	3309.4+563.6	3873.1	52(98)	1051.9
TPB8-99	40(40)	2.1+2.7	4.8	43(43)	2.1+2.9	5	41(52)	5.3
TPB8-199	48(48)	33.2+20.5	53.7	50(50)	35.4+20.6	56	47(74)	38.8
TPB8-299	66(106)	168.9+93.1	292	55(55)	195.3+72.8	268.2	52(90)	146.8
TPB8-399	101(156)	546.1+304.3	851.4	60(63)	637.0+174.6	808.7	58(105)	447.2
TPB8-499	133(195)	1404.1+745.6	2149.8	62(70)	1589.4+337.4	1926.9	57(105)	693.1
TPB8-599	167(364)	3033+1725.1	4758.1	65(76)	3595.4+622.9	4218.4	58(112)	1175.5

Table 7: Numerical results: boundary control

	IP-Hestenes			IP-PCG1			IP-PCG2	
Problem	it.(inn.)	time(prep.+iter.)	total	it.(inn.)	time(prep.+iter.)	total	it.(inn.)	total
TPB9-119	41(41)	4.4+4.3	8.7	45(47)	4.5+4.8	9.3	48(74)	9.6
TPB9-179	75(87)	17.1+25.7	42.8	51(56)	23.2+16.9	40.1	46(73)	30.1
TPB9-279	63(63)	133.1+69.4	202.5	57(63)	140.7+62.4	203.4	51(90)	136.2
TPB9-379	82(97)	448.9+212.5	661.4	62(78)	482.3+161.9	644.2	55(112)	302.1
TPB9-479	95(185)	1226.6+523.1	1749.7	65(84)	1255.2+341.1	1596.3	58(129)	638.5
TPB9-579	110(275)	2562+997.8	3560	67(96)	2658.7+570.9	3229.6	61(149)	1545.1
TPB10-119	44(44)	4.4+4.6	9	49(52)	4.5+5.2	9.8	44(70)	8.9
TPB10-179	56(56)	22.4+18.4	40.8	59(65)	23.2+19.5	42.7	55(89)	36
TPB10-279	72(87)	129.1+80.1	209.2	67(79)	140.9+74.3	215.2	63(117)	169.2
TPB10-379	101(251)	452.9+290.5	743.5	77(100)	483.6+204.9	688.6	74(165)	408.6
TPB10-479	105(360)	1202.8+629.1	1832	81(113)	1235.4+429.9	1665.3	78(190)	864.7
TPB10-579	119(374)	2558.6+1123.8	3682.5	86(130)	2660.6+738.9	2299.6	83(224)	2118.1

Table 8: Numerical results: boundary control

	IP-Hestenes			IP-PCG1			IP-PCG2	
Problem	it.(inn.)	time(prep.+iter.)	total	it.(inn.)	time(prep.+iter.)	total	it.(inn.)	total
TPD1-99	23(23)	4.8+2.2	7.1	26(25)	2.5+1.9	4.36	24(23)	3.3
TPD1-199	28(193)	123.1+26.4	149.5	28(26)	41.5+12.1	53.7	27(26)	22.6
TPD1-299	*	*	*	30(29)	218.3+41.2	259.5	28(27)	81.2
TPD1-399	*	*	*	31(56)	706.4+100.9	807.4	29(28)	222
TPD1-499	*	*	*	32(69)	2166.8+196.3	2363.2	29(28)	351.8
TPD2-99	28(28)	4.8+2.6	7.5	31(45)	2.5+2.4	4.9	29(28)	3.9
TPD2-199	31(166)	78.8+25.8	104.6	33(53)	41.7+15.7	57.4	30(29)	24.74
TPD2-299	*	*	*	34(64)	217.7+51.5	269.2	32(31)	92.8
TPD2-399	*	*	*	36(95)	704.9+125.7	830.7	33(32)	252.3
TPD2-499	*	*	*	37(132)	1741.9+252.1	1994.1	33(32)	399.1
TPD3-99	25(25)	4.8+2.4	7.2	31(26)	2.5+2.2 4.7		25(22)	3.4
TPD3-199	31(196)	119.0+27.9	147	33(27)	41.5+14.5	55.7	26(23)	21.7
TPD3-299	43(403)	694.4+131.1	825.6	34(28)	218.4+46.1	264.5	28(25)	80.8
TPD3-399	89(1184)	2339.9+758.3	3098.2	37(58)	869.4+119.4	988.9	30(27)	229.1
TPD3-499	*	*	*	36(61)	1742.25+212.91	1955.3	29(26)	350.23
TPD4-99	24(54)	5.0+2.9	7.9	20(16)	3.08+1.45	4.53	20(38)	3.11
TPD4-199	27(237)	80.6+29.4	109.9	21(17)	40.82+9.11	49.94	21(37)	19.02
TPD4-299	35(335)	424.2+108.3	532.5	22(18)	227.65+30.02	257.67	22(39)	68.21
TPD4-399	36(351)	1480.1+256.3	1736.5	23(19)	752.20+69.30	821.5	23(42)	184.77
TPD4-499	*	*	*	23(19)	2119.59+127.88	2247.47	23(42)	290.58

Table 9: Numerical results: distributed control

	IP-Hestenes			IP-PCG1			IP-PCG2	
Problem	it.(inn.)	time(prep.+iter.)	total	it.(inn.)	time(prep.+iter.)	total	it.(inn.)	total
TPD5-99	48(63)	5.0+4.8	9.9	56(152)	2.6+5.2	7.8	47(43)	6.4
TPD5-199	68(383)	80.7+58.2	138.9	78(712)	52.7+74.7	127.4	65(61)	53.9
TPD5-299	104(1439)	421.5+403.4	825.4	91(1356)	226.9+320.7	547.7	80(77)	230.9
TPD5-399	155(2255)	1489.0+1376.1	2865.2	107(1436)	718.6+704.4	1423.1	93(92)	709.7
TPD5-499	i	i	i	116(3125)	1798.6+2040.4	3839.1	104(104)	1256
TPD6-99	28(29)	5.77+2.7	8.48	35(70)	2.46+3.03	5.5	34(122)	6.28
TPD6-199	48(49)	118.03+25.11	143.17	51(88)	41.25+25.19	66.44	51(178)	53.2
TPD6-299	81(111)	686.30+131.49	817.99	56(97)	223.61+85.79	309.41	54(177)	173.82
TPD6-399	102(153)	2292.11+477.5	2769.7	71(130)	727.06+239.67	966.73	64(221)	553.78
TPD6-499	101(166)	5496.66+699.3	6196.11	62(107)	1849.82+361.12	2210.95	61(209)	823.08
TPD7-99	51(51)	4.8+4.9	9.7	51(90)	2.5+4.2	6.7	35(70)	5.5
TPD7-199	62(107)	118.7+35.6	154.3	63(284)	41.4+41.8	83.2	51(88)	45.8
TPD7-299	68(188)	684.8+127.4	812.4	70(493)	217.14+164.1	381.29	54(94)	158.7
TPD7-399	80(1010)	2299.4+654.9	2954.3	81(1014)	703.2+522.5	1225.8	65(109)	515.9
TPD7-499	90(1170)	3808.8+1150.7	4959.6	87(1331)	1733.9+1083.6	2817.7	80(115)	989.4

Table 10: Numerical results: distributed control

Table 11: Comparison PCG2 vs. KNITRO-3.1

	IP-PCG2	KNITRO-I	KNITRO-D
TPB1-99	6	40	17
TPB1-199	46	321	127
TPB1-299	243	1353	759
TPB1-399	799	4990	1939
TPB1-499	1372	10343	*

References

[1] Altman A. , Gondzio J.; *Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization*, Optim. Methods & Software 11–12 (1999) 275–302.

[2] Bergamaschi L., Gondzio J., Zilli G.; *Preconditioning indefinite systems in Interior Point methods for optimization*, Computational Optimization and Applications 28(2) (2004), 149–171.

[3] Bonettini S., Galligani E., Ruggiero V.; *An Inexact Newton method combined with Hestenes multipliers' scheme for the solution of Karush–Kuhn–Tucker systems*, to appear on Applied Math. Comput. (2004).

[4] Bunch J. R., Parlett B. N.;*Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal. 8 (1971) 639-655.

[5] Byrd R. H., Hribar M. E., Nocedal J.;*An Interior Point algorithm for large scale nonlinear programming*, SIAM J. Optim., 9(4) (1999), 877–900.

[6] D'Apuzzo M., Marino M.; *Parallel computational issues of an Interior–Point method for solving large bound constrained quadratic programming problems*, Parallel Computing, 29 (2003), 467–483.

[7] Durazzi C., Ruggiero V.; *A Newton Inexact Interior–Point method for large scale nonlinear optimization problems*, Annali Univ. Ferrara, Sez. VII, Sc. Matem. IL (2003), 333–357.

[8] Durazzi C., Ruggiero V.; *Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems*, Numer. Linear Algebra Appl., 10 (2003), 673-688 .

[9] Durazzi C., Ruggiero V., Zanghirati G.; *Parallel interior–point method for linear and quadratic programs with special structure*, J. Optim. Theory Appl. 110 (2001), 289–313.

[10] Harwell Subroutine Library; *A Catalogue of Subroutines (HSL 2000)*, AEA Technology, Harwell, Oxfordshire, England (2002).

[11] El–Bakry A.S., Tapia R.A., Tsuchiya T., Zhang Y.; *On the formulation and theory of Newton interior–point method for nonlinear programming*, J. Optim. Theory Appl. 89 (1996), 507–541.

[12] Gould N.I.M.; *On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem*, Math. Programming 32 (1985), 90–99.

[13] Hestenes M.R.; Optimization Theory. The Finite Dimensional Case, J. Wiley & Sons, New York, 1975.

[14] Liu J.W., Ng E.G., Peyton B.W.; *On finding supernodes for sparse matrix computations*, SIAM J. Matrix Anal. Appl. 14 (1993), 242–252.

[15] Luenberger D.G.; Linear and Nonlinear Programming, 2nd edition, Addison–Wesley, Reading MA, 1984.

[16] Lukšan L., Vlček J.; *Indefinitely preconditioned Inexact Newton method for large sparse equality constrained non–linear programming problems*, Numer. Linear Algebra Appl., 5 (1998), 219–247.

[17] Lukšan L., Matonoha C., Vlček J.; *Interior–Point method for nonlinear nonconvex optimization*, to appear on Numer. Linear Algebra Appl. (2004).

[18] Mittelmann, H.D. ; Private Communication (2004).

[19] Maurer H., Mittelmann H.D.; *Optimization techniques for solving elliptic control problems with control and state constraints: Part 1. Boundary control*, Comput. Optim. Appl. 16 (2000), 29–55.

[20] Maurer H., Mittelmann H.D.; *Optimization techniques for solving elliptic control problems with control and state constraints: Part 2. Distributed control*, Comput. Optim. Appl. 18 (2001), 141–160.

[21] Mittelmann H.D., Maurer H.; *Solving elliptic control problems with Interior Point and SQP Methods: control and state constraint*, J. Comput. Appl. Math. 120 (2000), 175–195.

[22] Nocedal J., Wright S.J.; Numerical Optimization, Springer, New York, 1999.

[23] Saad Y.; Iterative Methods for Sparse Linear System, PSW Publ. Co., Boston MA, 1996.

[24] Saunders M., Tomlin J. A., *Solving regularized linear programs using barrier methods and KKT systems*, Tech. Report SOL 96–4, Systems Optimization Laboratory, Dept. of Operations Research, Stanford University, Stanford, CA 94305, December 1996.

[25] Vanderbei R.J.; *Symmetric quasidefinite matrices*, SIAM J. Optim. 5 (1999) 100–113.

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