Analisi Numerica: Metodi e Software Matematico

Progetto cofinanziato dal MURST - 1998–2000

Projection–Type Methods for Large Convex Quadratic Programs: Theory and Computational Experience

V. Ruggiero

Dipartimento di Matematica
Università di Ferrara

E. Galligani L. Zanni

Dipartimento di Matematica
Università di Modena e Reggio Emilia

Monografia n. 5

Ferrara 2000

Monografie del Programma di Ricerca Scientifica 1998–2000
"Analisi Numerica: Metodi e Software Matematico"
cofinanziato dal Ministero per l'Università e
la Ricerca Scientifica e Tecnologica nell'anno 1997.

Monographs of the 1998–2000 Scientific Research Program
"Numerical Analysis: Methods and Mathematical Software"
supported in 1997 by the Italian Ministry for the University
and the Scientific and Technological Research.

http://www.unife.it/AnNum97

Abstract

A well-known approach to the solution of large and sparse linearly constrained quadratic programming (QP) problems is given by the classical projection and projection methods. These methods have a similar iterative scheme consisting in to solve a sequence of QP subproblems with the constraints of the original problem and an easily solvable Hessian matrix. A theorem of convergence, resuming the main results about the projection and splitting methods, is given for this general scheme.
In order to achieve an higher numerical performance than that obtained by the projection and splitting methods, we introduce two variants of an iterative scheme that use a variable projection parameter at each step. These two variable projection–type methods differ in the strategy used to assure a sufficient decrease in the objective function $f(x)$. We prove, under very general hypotheses, the convergence of these schemes and we propose two practical, nonexpensive and efficient updating rules for updating the projection paremeter.
An extensive numerical experimentation shows that the variable projection–type methods have an efficient behaviour.
These results are produced by a set of programs available at the URL:
http://www.unife.it/AnNum97/index2.html.
About the solution of the inner subproblems of the projection–type methods, we observe that, when the structure of the constraints does not suggest a particular solver for the QP inner subproblems, we can formulate each inner subproblem as a mixed linear complementarity problem (LCP) that can be solved by the classical projected SOR method of Cryer. When the number of constraints (and, consequently, the size of the inner LCPs) is large, appropriate parallel splitting methods for LCPs can be used for the solution on multiprocessor systems. Finally we describe two applications, arising in the framework of data analysis, that involve QP problems suited to be solved by projection-type methods.

V. Ruggiero
Dipartimento di Matematica, Università di Ferrara, Via Machiavelli 35, Ferrara, Italy.
E-mail: `rgv@dns.unife.it`.

E. Galligani L. Zanni
Dipartimento di Matematica, Università di Modena e Reggio Emilia, Via Campi 213/b, 41100 Modena, Italy. E-mail: {`galligan,zanniluca`}`@unimo.it`.

Contents

Contents **i**

1 Introduction **1**

2 Preliminary definitions and results **3**

3 The classical projection and splitting methods **5**

4 Variable Projection-Type Methods **11**
4.1 The Variable Projection Method 12
4.2 The Adaptive Variable Projection Method 16
4.3 Updating Rules for the Projection Parameter 18

5 Solution of the Inner QP Subproblems **21**
5.1 Parallel solvers for symmetric LCPs 22

6 Computational Experiments **27**
6.1 Numerical behaviour of the projection type methods for medium–scale and large–scale QP problems 28
6.2 Comparison among the VPM and the active set method for large–scale QP problems 31
6.3 Comparison among the scaled gradient projection methods and the variable projection–type methods 32
6.4 Numerical behaviour of parallel inner solvers in the projection–type methods 34

7 Two applications **39**
7.1 A Constrained Bivariate Interpolation Problem 39
7.2 Training Support Vector Machines 41

Appendix **47**

Bibliography **49**

Chapter 1

Introduction

Consider the linearly constrained convex quadratic programming (QP) problem

$$\begin{array}{ll} \textit{minimize} & f(x) = \frac{1}{2}x^TGx + q^Tx \\ \textit{subject to} & Cx = d, \qquad Ax \geq b, \end{array} \tag{1.1}$$

where G is a symmetric positive semidefinite matrix of order n, C is an $m_e \times n$ matrix of full row rank ($m_e \leq n$) and A is an $m_i \times n$ matrix. We assume that the feasible region $K = \{x \mid Cx = d,\ Ax \geq b\}$ is a nonempty polyhedral set and that $f(x)$ is bounded from below on K; then, the set of optimal solutions of (1.1), denoted by K^*, is also a nonempty polyhedral set, given by $K^* = \{x \in K \mid E(x - x^*) = 0,\ q^T(x - x^*) = 0\}$, where x^* is a solution of (1.1) and E is a matrix such that $G = E^TE$. For any $x \in K^*$, $f(x) = f^*$, where f^* denotes the optimal value of (1.1).
When the matrices G, C and A are large, sparse and without a particular structure, an approach for solving the problem (1.1) is represented by the classical projection and splitting methods [25], [44], that are largely developed also for linear complementarity and variational inequality problems. These last problems are related to (1.1); indeed, since $x^* \in K$ is a solution of (1.1) if and only if it satisfies the inequality

$$(Gx^* + q)^T(x - x^*) \geq 0, \qquad \text{for any} \qquad x \in K,$$

the problem (1.1) can be considered as a symmetric affine variational inequality (AVI) problem. Futhermore, when $K = \{x \mid x_i \geq 0; i \in \mathcal{I} \subseteq \{1, 2, ..., n\}\}$, the problem (1.1) is equivalent to the following symmetric monotone linear complementarity problem (LCP) ($\mathcal{I} = \{1, 2, ..., n\}$) or mixed LCP ($\mathcal{I} \subset \{1, 2, ..., n\}$)

$$\begin{gathered} u = Gx + q, \\ x_i \geq 0, \quad u_i \geq 0, \quad x_iu_i = 0, \quad i \in \mathcal{I}, \\ u_i = 0, \quad i \notin \mathcal{I}. \end{gathered}$$

After a preliminary chapter that collects a set of basic definitions and results, in Chapter 3 we show that the introduction of a scalar parameter, known as projection parameter, enables the classical projection and splitting methods to be unified in a common iterative scheme; furthermore, the main features and convergence results related to these methods are summarized.
In order to achieve an higher numerical performance than that obtained by the classical projection and splitting methods, in [46], [47] and [48] we introduced two variants of an iterative scheme that uses a variable scalar parameter at each step. These two methods, called variable projection method (VPM) and adaptive variable projection method

(AVPM), differ in the strategy used to assure a sufficient decrease in the objective function $f(x)$ and, consequently, the convergence of the schemes. In Chapter 4, we describe these methods and we propose two updating rules for the variable projection parameter. In Chapter 5, we discuss the solution of the inner subproblems.

In Chapter 6, we report the results of numerical experiments concerning the methods described in previous chapters. In Chapter 7, we describe two applications, arising in the framework of data analysis, that involve QP problems suited to be solved by projection–type methods. In Appendix, we report the references to a set of Fortran 77 programs used for the numerical experiments of Chapter 6.

In the sequel of the work, we denote by $\lambda_{min}(G)$ and $\lambda_{max}(G)$ the minimum and the maximum eigenvalue, respectively, of a symmetric matrix G. For any matrix G, G_S denotes the symmetric part $\frac{G+G^T}{2}$ of G. We indicate by $\|x\|$ the usual Euclidean norm of x and by $\|x\|_G$ the elliptic norm of x defined as $\sqrt{x^TGx}$ where G is a symmetric positive definite matrix. Furthermore, we denote by $\|B\|$ the norm of a matrix B induced by the vector Euclidean norm. Finally, we indicate by $[x]^+$ the orthogonal projection of a vector x onto a closed convex set K, i.e., $[x]^+ = \arg\min_{y\in K} \|y-x\|$, and by $\phi(x)$ the distance from x to K^*, i.e.,

$$\phi(x) = \min_{\overline{x}\in K^*} \|x-\overline{x}\|.$$

CHAPTER 2

Preliminary definitions and results

In this Chapter we recall some definitions and results that are basic for stating the convergence of the methods described in the next chapters.

LEMMA **1** [27] *Let G be a symmetric matrix. There exist two positive scalars η and τ such that*

$$\phi(x) \leq \tau\|x - [x - (Gx + q)]^+\| \tag{2.1}$$

for any $x \in K$, with $\|x - [x - (Gx + q)]^+\| \leq \eta$.

PROOF. See [27, pp. 45–46].

DEFINITION **2** is below; DEFINITION **1** *A sequence of nonzero vectors $\{x^k\}$ converges R–linearly to a vector $\overline{x}$ if*

$$\|x^k - \overline{x}\| \leq \alpha\gamma^k \quad \text{for any} \quad k \tag{2.2}$$

for some scalars $\alpha > 0$ and $\gamma \in (0, 1)$.

DEFINITION **2** *A sequence of nonzero vectors $\{x^k\}$ converges Q–linearly to a vector $\overline{x}$ if*

$$lim\, sup_{k\to\infty} \frac{\|x^{k+1} - \overline{x}\|}{\|x^k - \overline{x}\|} < 1. \tag{2.3}$$

For a detailed discussion about R–linear and Q–linear convergence, see [38].

LEMMA **2** [28] *Let $\{x^k\}$ be a sequence of vectors in K satisfying*

$$\|x^k - x^{k-1}\|^2 \leq \beta(f(x^{k-1}) - f(x^k)) \quad \textit{for any} \quad k \geq k_0 \tag{2.4}$$

for some scalar $\beta > 0$. If $\{f(x^k)\}$ converges R–linearly, then $\{x^k\}$ also converges al least R–linearly.

PROOF. We observe from (2.4) that, for $k \geq k_0$, $\{f(x^k)\}$ is a monotonically nonincreasing sequence. Since $f(x)$ is bounded from below on K, the sequence $\{f(x^k)\}$ is convergent as $k \to \infty$. We denote by $\overline{f}$ the limit point of the sequence. Since $\overline{f} \leq f(x^k)$, for any $k \geq k_0$, from (2.4) and (2.2), we can write

$$\|x^k - x^{k-1}\|^2 \leq \beta(f(x^{k-1}) - \overline{f}) \leq \beta\alpha\gamma^{k-1} \quad \text{for any} \quad k \geq k_0 \tag{2.5}$$

for some $\alpha > 0$ and $\gamma \in (0,1)$. Now, let k_1 be a positive integer such that

$$\overline{\gamma}^k < \frac{\epsilon(1-\overline{\gamma})}{\sqrt{\beta\alpha}} \quad \text{for any} \quad k \geq k_1. \tag{2.6}$$

for some $\epsilon > 0$ and $\overline{\gamma} = \sqrt{\gamma} \in (0,1)$. Then, from (2.5) and (2.6), for $k \geq \max(k_0, k_1)$ and any positive integer m, we have

$$\begin{aligned} \|x^{k+m} - x^k\| &= \left\| \sum_{i=1}^{m} x^{k+i} - x^{k+i-1} \right\| \\ &\leq \sum_{i=1}^{m} \|x^{k+i} - x^{k+i-1}\| \leq \sqrt{\beta\alpha} \sum_{i=1}^{m} \overline{\gamma}^{k+i-1} \\ &\leq \sqrt{\beta\alpha}\,\overline{\gamma}^k \frac{1-\overline{\gamma}^m}{1-\overline{\gamma}} \leq \epsilon. \end{aligned} \tag{2.7}$$

This shows that the $\{x^k\}$ is a Cauchy–sequence and, then, it is convergent. Let $\overline{x}$ be the limit point of the sequence $\{x^k\}$; as $m \to \infty$, the inequality (2.7) becomes

$$\|\overline{x} - x^k\| \leq \frac{\sqrt{\beta\alpha}}{1-\overline{\gamma}} \overline{\gamma}^k$$

for any $k \geq \max(k_0, k_1)$. Then $\{x^k\}$ converges to $\overline{x}$ at least R–linearly.

DEFINITION **3** *A splitting of a matrix G is a pair of matrices (D, H) such that $G = D + H$.*

DEFINITION **4** [37] *(D, H) is a P–regular splitting of the matrix G if D is a nonsingular matrix and $D - H$ is a positive definite matrix.*

LEMMA **3** *If G is a symmetric positive semidefinite matrix and (D, H) is a P–regular splitting of G, then D is a positive definite matrix and $\lambda_{max}(D_S^{-\frac{1}{2}} G D_S^{-\frac{1}{2}}) \in [0, 2)$.*

PROOF. Since $D = \frac{1}{2}G + \frac{1}{2}(D - H)$, D is a positive definite matrix. Furthermore we have that (D, H) is a P–regular splitting of G if and only if $(D - H)_S = D + D^T - G = 2D_S - G$ is a symmetric positive definite matrix. Then $0 < \lambda_{min}\left(D_S - \frac{1}{2}G\right) = \lambda_{min}\left(D_S^{\frac{1}{2}}\left(I - \frac{1}{2}D_S^{-\frac{1}{2}} G D_S^{-\frac{1}{2}}\right) D_S^{\frac{1}{2}}\right)$. Since $D_S^{\frac{1}{2}}\left(I - \frac{1}{2}D_S^{-\frac{1}{2}} G D_S^{-\frac{1}{2}}\right) D_S^{\frac{1}{2}}$ is congruent to $I - \frac{1}{2}D_S^{-\frac{1}{2}} G D_S^{-\frac{1}{2}}$, it follows

$$0 < \lambda_{min}\left(I - \frac{1}{2}D_S^{-\frac{1}{2}} G D_S^{-\frac{1}{2}}\right) = 1 - \lambda_{max}\left(\frac{1}{2}D_S^{-\frac{1}{2}} G D_S^{-\frac{1}{2}}\right). \tag{2.8}$$

Chapter 3

The classical projection and splitting methods

The classical projection and splitting methods have a common iterative scheme which consists in to generate, from an arbitrary vector x^0, a sequence of vectors $\{x^k\}$, $k \geq 1$, by solving the following QP subproblems

$$\text{(3.1)} \qquad \begin{array}{ll} \textit{minimize} & \frac{1}{2}x^T\frac{D}{\rho}x + (q + (G - \frac{D}{\rho})x^{k-1})^T x \\ \textit{subject to} & Cx = d, \qquad Ax \geq b, \end{array}$$

where ρ is a positive parameter, known as projection parameter, and D is a prefixed symmetric positive definite matrix of order n. From the practical point of view, the matrix D must be an easily solvable matrix (diagonal or block diagonal matrix).
When $\rho = 1$ in (3.1), we refer to the iterative scheme as a splitting method (SM) while, for $\rho \neq 1$, we have a projection method (PM).
The iterative scheme (3.1) appears convenient for the solution of large–scale sparse QP problems, because it allows any sparsity of the objective and constraint matrices to be readily exploited, uses little storage and is well suited to implementation on parallel computers. Indeed, each iteration consists essentially in computing the matrix–vector product $(G - \frac{D}{\rho})x^{k-1}$ and solving the QP subproblem (3.1). For this subproblem, according to the structure of the constraints of the application in question, we can use specialized sequential or parallel solvers for separable or nearly separable QP problems (see [36] and [41] as examples of solvers for separable quadratic programs with special constraints). When the constraints do not present a particular structure, we can formulate each inner subproblem as a mixed linear complementarity problem (LCP) that can be solved by sequential or parallel splitting methods (see Chapter 6).
In literature, there exists a vast collection of results regarding the convergence of the projection and splitting methods.
In the case of strictly convex QP problems, the convergence of the SM is obtained in [17] under the hypotheses that D is a symmetric matrix and $(D, G - D)$ is a P–regular splitting. In [18], using another technique in the proof of the convergence, we obtain that the SM converges at least Q–linearly.
In the case of convex QP problems, the R–linear convergence of the SM is proved in [28] under the hypotheses that D is a positive definite matrix and $(D, G - D)$ is a P–regular splitting of G. In this case, since D may be also an asymmetric matrix, x^k is the solution of the following AVI subproblem

$$\text{(3.2)} \qquad (Dx^k + (G - D)x^{k-1} + q)^T(x - x^k) \geq 0 \quad \text{for any} \quad x \in K.$$

When an iterative scheme is used for solving the inner subproblems (3.2), it is convenient to consider an inner progressive termination rule, in the sense that the accuracy in the solution of each subproblem depends on the quality of the previous iterate x^{k-1}: the closer x^{k-1} is to satisfying the external stopping criterion for the outer iteration, the more accurately the corresponding subproblem is solved. In this way, unnecessary inner iterations are avoided when x^{k-1} is far from the solution. Following [30], [28] and [24], this "inexact" solution of the AVI subproblem (3.2) can be viewed as "exact" solution of the perturbed AVI subproblem

$$(3.3)\qquad (Dx^k + (G-D)x^{k-1} + q + h^{k-1})^T(x - x^k) \geq 0 \quad \text{for any} \quad x \in K$$

where the "error" vector h^{k-1} satisfies the inequality:

$$(3.4)\qquad \|h^{k-1}\| \leq (\delta - \epsilon)\|x^k - x^{k-1}\|$$

with $\delta = \lambda_{min}(D_S - \frac{1}{2}G)$ and $\epsilon \in (0, \delta]$. The case $h^{k-1} = 0$ corresponds to an exact solution of (3.2). From the computational point of view, when K does not present a special structure, the AVI subproblem is generally more difficult than the QP subproblem (3.1). Possible exceptions are the symmetric monotone linear complementarity problems and the QP problems where $m_i = 0$, $x^TGx > 0$ for any x such that $Cx = 0$ and $m_e << n$. In this last case, each subproblem can be formulated [12] as a small–scale linear system of the form

$$(3.5)\qquad CD^{-1}C^T\lambda = d + D^{-1}C((G-D)x^{k-1} + q)$$

and it can be solved by a direct method. Nevertheless, when the system (3.5) is of medium–scale or large–scale, the choice of a symmetric matrix D enables a more efficient conjugate gradient method to be used for its solution.
The convergence of the PM with D symmetric positive definite can be immediately derived from the projection schemes for the variational inequality problems. In the case of strictly convex QP problems, by proceeding as in [10], the convergence of $\{x^k\}$ can be obtained by a contraction argument for $\rho < \frac{2\lambda_{min}(G)}{\lambda_{max}(GD^{-1}G)}$; the method has a Q–linear convergence with rate $(1 - \frac{\rho}{\lambda_{max}(D)}(2\lambda_{min}(G) - \rho\lambda_{max}(GD^{-1}G)))^{\frac{1}{2}}$. For convex QP problems, the convergence of the PM is proved under the sufficient condition $\rho < \frac{2\lambda_{min}(D)}{\lambda_{max}(G)}$ [33].
All the discussed projection and splitting methods can be included in a general iterative scheme, consisting in to generate, from an arbitrary point x^0, a sequence of vectors $\{x^k\}$, $k \geq 1$, where x^k is the solution of the following AVI subproblem:

$$(3.6)\qquad \left(\frac{D}{\rho}x^k + \left(G - \frac{D}{\rho}\right)x^{k-1} + q + h^{k-1}\right)^T (x - x^k) \geq 0 \text{ for any } x \in K$$

where $\rho > 0$, D is a positive definite matrix and h^{k-1} satisfies the inequality:

$$(3.7)\qquad \|h^{k-1}\|_{D_S^{-1}} \leq (\delta - \epsilon)\|x^k - x^{k-1}\|_{D_S}$$

with $\delta = \frac{1}{\rho} - \frac{1}{2}\lambda_{max}(D_S^{-\frac{1}{2}}GD_S^{-\frac{1}{2}})$ and $\epsilon \in (0, \delta]$. The following Theorem state the R–linear convergence of the scheme (3.6)–(3.6), resuming the main convergence results of the projection and splitting methods.

THEOREM **1** *Let the problem (1.1) be feasible with G symmetric positive semidefinite matrix and $f(x)$ bounded from below on K. Under one of the following condition:*

- $\rho = 1$ *and* $(D, G - D)$ *is a P–regular splitting of* G,
- D *positive definite matrix and* $0 < \rho < \frac{2}{\lambda_{max}(D_S^{-\frac{1}{2}} G D_S^{-\frac{1}{2}})}$,

the sequence of vectors $\{x^k\}$ *generated by the iterative scheme (3.6)–(3.7) is well-defined and converges to a solution of (1.1) at least R–linearly.*

PROOF. Since D is a positive definite matrix and $\rho > 0$, the AVI subproblem (3.6) has an unique solution x^k; then, for $k \geq 1$, $\{x^k\}$ is a well–defined sequence of feasible points of K.
Now, we consider the following equality

$$\begin{aligned} f(x^k) - f(x^{k-1}) &= (Gx^{k-1} + q)^T (x^k - x^{k-1}) \\ &\quad + \frac{1}{2}(x^k - x^{k-1})^T G (x^k - x^{k-1}). \end{aligned} \tag{3.8}$$

Since x^k is a solution of (3.6), from (3.6) computed at $x = x^{k-1}$ and (3.8), we can write

$$\begin{aligned} f(x^k) - f(x^{k-1}) &\leq -(x^k - x^{k-1})^T \left(\frac{D_S}{\rho} - \frac{1}{2}G\right)(x^k - x^{k-1}) \\ &\quad - {h^{k-1}}^T (x^k - x^{k-1}) \end{aligned} \tag{3.9}$$

From (3.7) and the hypotheses on D and ρ, we have

$$\begin{aligned} f(x^k) - f(x^{k-1}) &\leq \left(-\lambda_{min}\left(\frac{I}{\rho} - \frac{1}{2} D_S^{-\frac{1}{2}} G D_S^{-\frac{1}{2}}\right) + \delta - \epsilon\right) \|x^k - x^{k-1}\|^2_{D_S} \\ &\leq \left(-\frac{1}{\rho} + \frac{1}{2}\lambda_{max}(D_S^{-\frac{1}{2}} G D_S^{-\frac{1}{2}}) + \delta - \epsilon\right) \|x^k - x^{k-1}\|^2_{D_S} \\ &\leq -\alpha_1 \|x^k - x^{k-1}\|^2 \leq 0, \end{aligned} \tag{3.10}$$

where $\alpha_1 = \epsilon \lambda_{min}(D_S)$ and the equality to 0 holds when $x^{k-1} = x^k \in K^*$.
Then $\{f(x^k)\}$ is a sequence monotonically nonincreasing and, since $f(x)$ is bounded from below on K, it is convergent as $k \to \infty$ and $\lim_{k\to\infty} f(x^k) \geq f^*$. Thus, from (3.10), the sequence $\{x^k - x^{k-1}\}$ converges to the null vector as $k \to \infty$.
Now, we observe that the solution x^k of the subproblem (3.6) can be considered as the orthogonal projection of an appropriate vector onto K, i.e.

$$x^k = \left[x^k - \frac{D}{\rho}(x^k - x^{k-1}) - (Gx^{k-1} + q + h^{k-1})\right]^+. \tag{3.11}$$

From the nonexpansive property of the projection operator and from $\|h^{k-1}\| \leq (\delta - \epsilon)\lambda_{max}(D_S)\|x^k - x^{k-1}\|$ (see (3.7)), we have

$$\begin{aligned} \left\|x^{k-1} - [x^{k-1} - (Gx^{k-1} + q)]^+\right\| &= \Big\|x^{k-1} - [x^{k-1} - (Gx^{k-1} + q)]^+ + \\ &\quad - x^k + [x^k - \frac{D}{\rho}(x^k - x^{k-1}) - (Gx^{k-1} + q + h^{k-1})]^+\Big\| \\ &\leq \alpha_2 \|x^k - x^{k-1}\| \end{aligned} \tag{3.12}$$

with $\alpha_2 = \left(2 + \frac{\|D\|}{\rho} + (\delta - \epsilon)\lambda_{max}(D_S)\right)$. Since $\{\|x^k - x^{k-1}\|\}$ converges to 0 as $k \to \infty$, the sequence $\{\left\|x^{k-1} - [x^{k-1} - (Gx^{k-1} + q)]^+\right\|\}$ also converges to 0 as $k \to \infty$. Then, given a positive scalar α_3, there exists an integer $\overline{k} > 0$ such that, for any $k > \overline{k}$, we have

$$\left\|x^{k-1} - [x^{k-1} - (Gx^{k-1} + q)]^+\right\| \leq \alpha_3 \tag{3.13}$$

Hence, by Lemma 1, there exists a scalar $\alpha_4 > 0$ such that

$$\phi(x^{k-1}) \le \alpha_4 \left\| x^{k-1} - [x^{k-1} - (Gx^{k-1} + q)]^+ \right\| \quad \text{for any} \quad k > \overline{k}. \tag{3.14}$$

From (3.12), for any $k > \overline{k}$, we have

$$\phi(x^{k-1}) \le \alpha_5 \|x^k - x^{k-1}\| \tag{3.15}$$

with $\alpha_5 = \alpha_4 \cdot \alpha_2$; since $\{\|x^k - x^{k-1}\|\}$ converges to 0 as $k \to \infty$, the sequence $\{\phi(x^{k-1})\}$ converges to 0 as $k \to \infty$.
Let x^* the element of K^* closest to x^{k-1}, i.e. $\phi(x^{k-1}) = \|x^{k-1} - x^*\|$. From (3.6) computed at $x = x^*$ and the Mean Value Theorem, it is possible to write

$$\begin{aligned} \left(\frac{D}{\rho}(x^k - x^{k-1}) + Gx^{k-1} + q + h^{k-1}\right)^T (x^* - x^k) \quad &\ge \quad 0, \\ f(x^k) - f(x^*) = (G\zeta^k + q)^T (x^k - x^*), & \end{aligned}$$

where ζ^k lies on the line segment joining x^k with x^*. Thus, combining the above relations, we have

$$\begin{aligned} f(x^k) - f^* &= f(x^k) - f(x^*) \\ &\le \left(\lambda_{max}(G)\|\zeta^k - x^{k-1}\| + \|h^{k-1}\| + \frac{\|D\|}{\rho}\|x^k - x^{k-1}\|\right) \|x^k - x^*\|. \end{aligned}$$

Now the following inequalities hold:

$$\|\zeta^k - x^{k-1}\| \le \|x^k - x^{k-1}\| + \|\zeta^k - x^k\|,$$

$$\|\zeta^k - x^{k-1}\| \le \|x^{k-1} - x^*\| + \|x^* - \zeta^k\|,$$

and

$$\|x^k - x^*\| = \|\zeta^k - x^k\| + \|x^* - \zeta^k\|.$$

By adding the first two inequalities and using the last equality, we obtain

$$\|\zeta^k - x^{k-1}\| \le \|x^k - x^{k-1}\| + \|x^{k-1} - x^*\| = \|x^k - x^{k-1}\| + \phi(x^{k-1}).$$

From the above inequality and from $\|x^k - x^*\| \le \|x^k - x^{k-1}\| + \phi(x^{k-1})$, we obtain

$$\begin{aligned} f(x^k) \ - \ f^* \le & \\ & \big(\lambda_{max}(G)(\|x^k - x^{k-1}\| + \phi(x^{k-1})) + (\delta - \epsilon)\lambda_{max}(D_S)\|x^k - x^{k-1}\| + \\ & \left.\frac{\|D\|}{\rho}\|x^k - x^{k-1}\|\right) (\|x^k - x^{k-1}\| + \phi(x^{k-1})). \end{aligned}$$

From (3.15), for $k > \overline{k}$, we can write

$$f(x^k) - f^* \le \alpha_6 \|x^k - x^{k-1}\|^2$$

with $\alpha_6 = \left(\lambda_{max}(G)(\alpha_5 + 1) + (\delta - \epsilon)\lambda_{max}(D_S) + \frac{\|D\|)}{\rho}\right)(\alpha_5 + 1)$. Then, from (3.10), we have

$$f(x^k) - f^* \le \frac{\alpha_6}{\alpha_1}(f(x^{k-1}) - f(x^k)), \qquad \text{for any} \quad k > \overline{k}. \tag{3.16}$$

By rearranging terms in (3.16), we obtain

$$f(x^k) - f^* \le \frac{\alpha_6}{\alpha_6 + \alpha_1}(f(x^{k-1}) - f^*), \qquad \text{for any} \quad k > \overline{k},$$

and then $\{f(x^k)\}$ converges at least linearly to f^*. Since (3.10) holds, it follows from Lemma 2 that $\{x^k\}$ converges at least linearly. Since $\{\phi(x^k)\}$ converges to zero as $k \to \infty$, the limit point of $\{x^k\}$ is an element of K^*.

In general, D is chosen as a symmetric matrix and the subproblems (3.6) become strictly convex QP problems of the form (3.1). In this case, under the assumption that (3.1) are exactly solved, the assertions of Theorem 1 can be also proved by different techniques (see [47]).
From the viewpoint of the numerical effectiveness, in the splitting methods the practical requirement to select an easily solvable matrix D is very often in conflict with the choice of a splitting of G that implies fast convergence (unless the splitting is suggested by the underlying application; see, for example, Table 14 in Chapter 6, where we show the results of numerical experiments concerning the quadratic program arising in the constrained bivariate interpolation problem [16], [17]). In [18] an acceleration technique is proposed, consisting in computing the iterate x^k by the formula:

$$x^k = x^{k-1} + \theta_k(y^k - x^{k-1}) \tag{3.17}$$

where y^k denotes the solution of the subproblem (3.1) and $\theta_k = \arg\min_{\theta\in(0,1]} f(x^{k-1} + \theta(y^k - x^{k-1}))$. This correction step does not introduce additional matrix–vector products (see Chapter 4 for details) and the numerical experiments show that the reduction in the number of iterations is equal at most to 10%.

On the other hand, the projection schemes allow any symmetric positive definite matrix D to be selected, but the values of ρ that satisfy the sufficient convergence conditions are often so small as to determine very slow convergence. Furthermore, in some cases, it is possible to find values of ρ that do not satisfy the sufficient convergence condition and imply a fast convergence (see Chapter 6).
The drawbacks of the classical splitting and projection methods are avoided in the modified projection–type methods by introducing a correction step after the solution of each QP subproblems. One of the most general modified projection–type methods (MPM) is the scheme proposed in [49] in the context of the solution of monotone affine variational inequality problems. Starting from an arbitrary vector x^0, this scheme consists in to generate a sequence $\{x^k\}$, where x^k, $k = 1, 2, ...$, is obtained by a projection step that computes the solution y^k of the following QP subproblem:

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}x^Tx + (q + (G - I)x^{k-1})^Tx \\ \textit{subject to} & Cx = d, \qquad Ax \geq b, \end{array} \tag{3.18}$$

followed by the correction formula:

$$x^k = x^{k-1} + \gamma_{k-1}P^{-1}(I + G)(y^k - x^{k-1}) \tag{3.19}$$

with

$$\gamma_{k-1} = \theta \frac{\|y^k - x^{k-1}\|^2}{\|P^{-\frac{1}{2}}(I + G)(y^k - x^{k-1})\|^2}. \tag{3.20}$$

Here P is a symmetric positive definite matrix and θ is a parameter in the interval $(0, 2)$. When $\theta = 1$ and $P = I + G$, we have the projection and contraction method proposed in [21]. For appropriate choices of P (e.g. $P = I + G$), the correction step does not introduce additional matrix–vector products. The performance of this scheme is strongly dependent on the choice of P, θ and on the scaling of G and q, as emphasized also in [49].

CHAPTER 4

Variable Projection-Type Methods

In the framework of the QP problems, the classical projection and splitting methods are known also as scaled gradient projection methods "with constant stepsize" [4, p. 207]. In order to avoid the difficulty in finding a convenient value for the projection parameter, a limited minimization rule is introduced in the iterative scheme, giving rise to a scaled gradient projection method with a "limited minimization rule"; indeed, (as in the MPM) the iterate x^k is obtained by the correction formula:

$$x^k = x^{k-1} + \theta_k(y^k - x^{k-1}), \tag{4.1}$$

where y^k denotes the solution of (3.1) and $\theta_k = \arg\min_{\theta\in(0,1]} f(x^{k-1} + \theta(y^k - x^{k-1}))$. A convenient value for the projection parameter can also be obtained by making a search along the projection arc (scaled gradient projection method with an "Armijo rule along a projection arc" [4]). In particular, given $\overline{\rho} > 0$ and denoted by $x^k(\rho)$ the solution of the subproblem (3.1), the iterate x^k is equal to $x^k(\beta^m\overline{\rho})$ where m is the first nonnegative integer for which the following Armijo-like inequality holds:

$$f(x^{k-1}) - f\left(x^k(\beta^m\overline{\rho})\right) \geq \sigma\nabla f(x^{k-1})^T\left(x^{k-1} - x^k(\beta^m\overline{\rho})\right),$$

where $\beta \in (0,1)$ and $\sigma \in (0,1)$ are prefixed scalars.
Another way to improve the projection methods consists in using, at each iteration, a variable projection parameter; on this idea is based the variable projection method (VPM), introduced in [46] for strictly convex QP problems and in [47], [48] for convex QP problem. The projection parameter is updated according to a nonexpensive rule, deduced by heuristic considerations, and the convergence of the method is obtained under weak hypotheses by combining the projection step with a correction rule of the form (4.1), as in the scaled gradient projection methods with a "limited minimization rule".
In [47] and [48] we also introduce a variant of the VPM, named adaptive variable projection method (AVPM). In this last method, the value of the projection parameter given by an updating rule is adaptively reduced until the projection step produces a sufficient decrease in the objective function. In practice, the VPM and the AVPM generalize the above scaled gradient projection schemes with a "limited minimization rule" and with an "Armijo rule along a projection arc", by introducing a rule that determines a convenient value for ρ at each step.
In the following we describe the VPM and the AVPM and we recall Theorems on the R–linear convergence of these schemes.

4.1 The Variable Projection Method

Given a symmetric positive definite matrix D, the VPM for the solution of (1.1) can be stated as follows:

1. Let x^0 be an arbitrary vector and ρ_1 be an arbitrary positive constant; $k \leftarrow 1$;
2. Compute the unique solution y^k of the subproblem

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}x^T \frac{D}{\rho_k} x + (q + h^{k-1} + (G - \frac{D}{\rho_k})x^{k-1})^T x \\ \textit{subject to} & Cx = d, \qquad Ax \geq b; \end{array} \tag{4.2}$$

3. Set $d^k = y^k - x^{k-1}$;
4. If $(Gx^{k-1} + q)^T d^k < 0$ and $k \neq 1$, compute the solution θ_k of the problem

$$\min\{f(x^{k-1} + \theta d^k); \quad \theta \in (0, 1]\}; \tag{4.3}$$

   else

$$\theta_k = 1;$$

5. Compute

$$x^k = x^{k-1} + \theta_k d^k; \tag{4.4}$$

6. Terminate if x^k satisfies a stopping rule, otherwise update ρ_{k+1} by an appropriate rule; then $k \leftarrow k + 1$ and go to step 2.

The aim of the initial step of the scheme is to generate, from an arbitrary point x^0, a vector $x^1 \in K$. When $\lambda_{min}(D)$ is easily computable, a convenient value for ρ_1 is $\rho_1 = \lambda_{min}(D) / \left(\sum_{i=1}^n \sum_{j=1}^n g_{ij}^2 \right)^{1/2}$, where g_{ij} is the ij-th element of G; this value satisfies the sufficient convergence condition for the step size of the projection methods and guarantees that $\|x^1 - x^*\|_D \leq \|x^0 - x^*\|_D$, with $x^* \in K^*$.
When the computation of Dx^{k-1} is nonexpensive, the computational complexity of one iteration of the VPM is essentially equal to that of the projection methods. Indeed, if we keep stored from the previous iteration the vector $t = Gx^{k-1}$, at the k-th iteration the computation of θ_k requires the matrix–vector product $\bar{t} = Gy^k$ and some less expensive vector operations. Now, the vector t is updated by the following rule:

$$t \leftarrow t + \theta_k(\bar{t} - t).$$

We will prove the convergence of the VPM under the following assumptions:

AS1. G is a symmetric positive semidefinite matrix, K is a nonempty set and $f(x)$ is bounded from below on K;

AS2. D is a symmetric positive definite matrix and the sequence of scalars $\{\rho_k\}$ is bounded from below and above by positive constants: $0 < \rho_m \leq \rho_k \leq \rho_M$;

AS3. the sequence of the error vectors $\{h^k\}$ satisfies the following condition:

$$\|h^{k-1}\| \le \left(\frac{\lambda_{min}(D)}{\rho_k} - \epsilon_k\right)\|y^k - x^{k-1}\| \qquad k > 1 \tag{4.5}$$

with $\epsilon_k \in [\frac{\lambda_{min}(D)}{\rho_M}, \frac{\lambda_{min}(D)}{\rho_k}]$.
The assumption AS2 on $\{\rho_k\}$ is very general; the choice of a particular bounded sequence $\{\rho_k\}$ is based only on its numerical effectiveness. The case of a constant projection parameter is a special case of this.
The following Lemma states some basic assertions for the convergence of the VPM.

LEMMA **4** *If, for a given $k > 1$, x^{k-1} is feasible to (1.1) and $x^{k-1} \neq y^k$, then the following assertions hold:*

a. *y^k is well–defined and is a feasible point of (1.1); furthermore, d^k is a descent direction for the objective function $f(x)$ in x^{k-1};*

b. *the sequence $\{\theta_k\}$ satisfies the following bounds:*

$$0 < \theta_{min} = \min\left\{1, \frac{\lambda_{min}(D)}{\rho_M \lambda_{max}(G)}\right\} \le \theta_k \le 1, \qquad \text{for any } k > 1. \tag{4.6}$$

PROOF. **a.** For the uniqueness of the solution of the linearly constrained strictly convex subproblem (4.2), y^k is well-defined and it is a feasible point of (1.1). Furthermore, we consider the first order optimality condition for the solution y^k of the subproblem (4.2):

$$\left(\frac{D}{\rho_k}(y^k - x^{k-1}) + Gx^{k-1} + q + h^{k-1}\right)^T (x - y^k) \ge 0, \quad \text{for any } x \in K. \tag{4.7}$$

Taking $x = x^{k-1}$ in (4.7), since $\frac{D}{\rho_k}$ is a symmetric positive definite matrix and $x^{k-1} \neq y^k$, we obtain from (4.5) that d^k is a descent direction for $f(x)$ in x^{k-1}:

$$\begin{aligned}(Gx^{k-1} + q)^T d^k &\le -{d^k}^T \frac{D}{\rho_k} d^k - {h^{k-1}}^T d^k \\ &\le \left(-\frac{\lambda_{min}(D)}{\rho_k} + \frac{\lambda_{min}(D)}{\rho_k} - \epsilon_k\right)\|d^k\|^2 \\ &\le -\frac{\lambda_{min}(D)}{\rho_M}\|d^k\|^2 < 0.\end{aligned} \tag{4.8}$$

When $x^{k-1} = y^k$, then $x^{k-1} \in K^*$.
b. For $k > 1$, the exact solution of the problem (4.3) is given by the following rule:

$$\theta_k = \begin{cases} \min\left\{-\frac{(Gx^{k-1}+q)^T d^k}{{d^k}^T G d^k}, 1\right\} & \text{if } {d^k}^T G d^k \neq 0 \\ 1 & \text{otherwise.}\end{cases} \tag{4.9}$$

From (4.8) and (4.9) the assertion **b** immediately follows.

THEOREM **2** *The sequence $\{x^k\}$, $k \ge 1$, generated by the VPM is a well-defined sequence of vectors of K and it is convergent to a solution of (1.1) at least R–linearly.*

PROOF. Since $x^{(1)} \in K$, from the assertions of Lemma 4 and the convexity of K, it follows that $\{x^k\}$ is a well-defined sequence of feasible points of (1.1) for $k \ge 1$.

From (4.3), (4.4) and the following equality:

$$f(x^{k-1}+\theta d^k)-f(x^{k-1})=\theta(Gx^{k-1}+q)^T d^k+\frac{1}{2}\theta^2 {d^k}^T G d^k, \tag{4.10}$$

we have

$$f(x^k)\le f(x^{k-1})+\theta(Gx^{k-1}+q)^T d^k+\frac{1}{2}\theta^2 {d^k}^T G d^k, \tag{4.11}$$

for any $\theta\in(0,1]$. If $d^{kT}Gd^k=0$ or $-\frac{(Gx^{k-1}+q)^T d^k}{d^{kT}Gd^k}\ge 1$, the solution of the problem (4.3) is obtained for $\theta_k=1$; from (4.6), (4.8) and (4.11) we have

$$\begin{aligned} f(x^k) &\le f(x^{k-1})+\frac{1}{2}(Gx^{k-1}+q)^T d^k \\ &\le f(x^{k-1})-\frac{\lambda_{min}(D)}{2\rho_M}\|d^k\|^2 \\ &\le f(x^{k-1})-\frac{\lambda_{min}(D)\theta_{min}}{2\rho_M}\|d^k\|^2. \end{aligned} \tag{4.12}$$

If $d^{kT}Gd^k\ne 0$ and $-\frac{(Gx^{k-1}+q)^T d^k}{d^{kT}Gd^k}<1$, the solution of the problem (4.3) is obtained for $\theta_k=-\frac{(Gx^{k-1}+q)^T d^k}{d^{kT}Gd^k}$; from (4.6), (4.8) and (4.11) we have

$$\begin{aligned} f(x^k) &\le f(x^{k-1})-\frac{1}{2}\frac{((Gx^{k-1}+q)^T d^k)^2}{d^{kT}Gd^k} \\ &= f(x^{k-1})+\frac{\theta_k}{2}(Gx^{k-1}+q)^T d^k \\ &\le f(x^{k-1})-\frac{\lambda_{min}(D)\theta_{min}}{2\rho_M}\|d^k\|^2. \end{aligned} \tag{4.13}$$

Thus, if $\alpha_1=\frac{\lambda_{min}(D)\theta_{min}}{2\rho_M}$, from (4.12), (4.13) and (4.4), we conclude that

$$\begin{aligned} f(x^{k-1})-f(x^k) &\ge \alpha_1\|d^k\|^2=\frac{\alpha_1}{\theta_k^2}\|x^k-x^{k-1}\|^2 \\ &\ge \alpha_1\|x^k-x^{k-1}\|^2\ge 0, \end{aligned} \tag{4.14}$$

where the equality to zero holds when $x^{k-1}=y^k=x^k\in K^*$.
Since $f(x)$ is bounded from below on K, the sequence $\{f(x^k)\}$ is convergent and $\lim_{k\to\infty}f(x^k)\ge f^*$. Thus, from (4.14) we have that the sequences $\{d^k\}$ and $\{x^k-x^{k-1}\}$ converge to the null vector as $k\to\infty$.
Now, we observe that the solution y^k of the subproblem (4.2) can be considered as the orthogonal projection of an appropriate vector onto K, i.e.

$$y^k=\left[y^k-\frac{D}{\rho_k}(y^k-x^{k-1})-(Gx^{k-1}+q+h^{k-1})\right]^+.$$

From the nonexpansive property of the projection operator, we have

$$\begin{aligned} \left\|x^{k-1}-[x^{k-1}-(Gx^{k-1}+q)]^+\right\| &= \Big\|x^{k-1}-[x^{k-1}-(Gx^{k-1}+q)]^+ + \\ &\quad -y^k+[y^k-\frac{D}{\rho_k}(y^k-x^{k-1})-(Gx^{k-1}+q+h^{k-1})]^+\Big\| \\ &\le \alpha_2\|d^k\| \end{aligned} \tag{4.15}$$

with $\alpha_2 = \left(2 + \frac{\lambda_{max}(D)+\lambda_{min}(D)}{\rho_m}\right)$. Since the sequence $\{\|d^k\|\}$ converges to 0 as $k \to \infty$, the sequence $\{\|x^{k-1} - [x^{k-1} - (Gx^{k-1} + q)]^+\|\}$ also converges to 0 as $k \to \infty$. Then, given a positive scalar α_3, there exists an integer $\overline{k} > 0$ such that, for any $k > \overline{k}$, we have

$$\|x^{k-1} - [x^{k-1} - (Gx^{k-1} + q)]^+\| \le \alpha_3.$$

Hence, by Lemma 1, there exists a scalar $\alpha_4 > 0$ such that:

$$\phi(x^{k-1}) \le \alpha_4 \|x^{k-1} - [x^{k-1} - (Gx^{k-1} + q)]^+\| \qquad \text{for any } k > \overline{k}$$

From (4.15), for any $k > \overline{k}$, we have

$$\phi(x^{k-1}) \le \alpha_5 \|d^k\| \tag{4.16}$$

where $\alpha_5 = \alpha_4 \cdot \alpha_2$; since $\{\|d^k\|\}$ converges to 0, the sequence $\{\phi(x^{k-1})\}$ also converges to 0 as $k \to \infty$.

Let x^* be the element of K^* closest to x^{k-1}, i.e. $\phi(x^{k-1}) = \|x^{k-1} - x^*\|$. From the first order optimality conditions for the problems (4.2) and the Mean Value Theorem, it is possible to write:

$$\begin{aligned} \left(\frac{D}{\rho_k}(y^k - x^{k-1}) + Gx^{k-1} + q + h^{k-1}\right)^T (x^* - y^k) \quad &\ge \quad 0, \\ f(y^k) - f(x^*) = (G\zeta^k + q)^T (y^k - x^*), \end{aligned}$$

where ζ^k lies on the line segment joining y^k with x^*. Thus, combining the above relations, we have

$$\begin{aligned} f(y^k) \quad - \quad & f^* = f(y^k) - f(x^*) \\ & \le \left(\lambda_{max}(G)\|\zeta^k - x^{k-1}\| + \|h^{k-1}\| + \frac{\lambda_{max}(D)}{\rho_m}\|d^k\|\right) \|y^k - x^*\|. \end{aligned}$$

Since $\|\zeta^k - x^{k-1}\| \le \phi(x^{k-1}) + \|d^k\|$ and $\|y^k - x^*\| \le \phi(x^{k-1}) + \|d^k\|$, from (4.5), we obtain

$$\begin{aligned} f(y^k) - f^* \quad \le \quad & \big(\lambda_{max}(G)\left(\phi(x^{k-1}) + \|d^k\|\right) \\ & + \frac{\lambda_{min}(D) + \lambda_{max}(D)}{\rho_m}\|d^k\|\Big) \left(\phi(x^{k-1}) + \|d^k\|\right). \end{aligned}$$

From (4.16), for any $k > \overline{k}$, we can write

$$f(y^k) - f^* \le \alpha_6 \|d^k\|^2$$

where $\alpha_6 = \left(\lambda_{max}(G)(\alpha_5 + 1) + \frac{\lambda_{min}(D)+\lambda_{max}(D)}{\rho_m}\right)(\alpha_5 + 1)$. Then, from (4.3), (4.4) and (4.14), we have

$$f(x^k) - f^* \le \frac{\alpha_6}{\alpha_1}(f(x^{k-1}) - f(x^k)), \qquad \forall\ k > \overline{k}. \tag{4.17}$$

By rearranging terms in (4.17), we obtain

$$f(x^k) - f^* \le \frac{\alpha_6}{\alpha_6 + \alpha_1}(f(x^{k-1}) - f^*), \qquad \forall\ k > \overline{k},$$

and then $\{f(x^k)\}$ converges at least linearly to f^*. Since (4.14) holds, then it follows from Lemma 2 that $\{x^k\}$ converges at least linearly. Since $\{\phi(x^k)\}$ converges to zero as $k \to \infty$, the limit point of $\{x^k\}$ is an element of K^*.

4.2 The Adaptive Variable Projection Method

Given a symmetric positive definite matrix D and two positive scalars $\beta \in (0,1)$ and $\eta > \frac{1}{2}$, the AVPM for the solution of (1.1) is stated as follows:

1. Let x^0 be an arbitrary vector and $\overline{\rho}_1$ be an arbitrary positive constant, $k \leftarrow 1$, $\rho_k = \overline{\rho}_1$, $m_k = 0$;
2. Compute the unique solution $x(\rho_k)$ of the subproblem

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}x^T \frac{D}{\rho_k} x + (q + h(\rho_k) + (G - \frac{D}{\rho_k})x^{k-1})^T x \\ \textit{subject to} & Cx = d, \qquad Ax \geq b; \end{array} \tag{4.18}$$

3. Set $d^k = x(\rho_k) - x^{k-1}$;
4. If $-(Gx^{k-1} + q)^T d^k < \eta d^{k^T} G d^k$ and $k \neq 1$, then $m_k = m_k + 1, \rho_k = \beta^{m_k}\overline{\rho}_k$ and go to step 2, otherwise $x^k = x(\rho_k)$;
5. Terminate if x^k satisfies a stopping rule, otherwise update ρ_{k+1} by an appropriate rule; then $\overline{\rho}_{k+1} = \rho_{k+1}$, $m_{k+1} = 0$, $k \leftarrow k+1$ and go to step 2.

The computational complexity of each iteration of the AVPM depends on the number $m_k + 1$ of inner subproblems that have to be solved to find a convenient value for ρ_k. Since the test at the step 4 requires the matrix–vector product $Gx(\rho_k)$, each iteration of the AVPM is essentially equivalent to $m_k + 1$ iterations of the projection methods.
As in the iterative scheme of the VPM, the aim of the first step is to find an initial feasible point x^1.
We will prove the convergence of the AVPM under the assumptions AS1 and AS2 for the VPM and the following condition:

AS4. the sequence of the error vectors $\{h(\rho_k)\}$ is such that

$$\|h(\rho_k)\| \leq \min\left(\eta\lambda_{max}(G), \left(\frac{\lambda_{min}(D)}{\rho_k} - \epsilon_k\right)\right) \|x(\rho_k) - x^{k-1}\| \tag{4.19}$$

$$k > 1, \ \text{with } \epsilon_k \in [\frac{\lambda_{min}(D)}{\rho_M}, \frac{\lambda_{min}(D)}{\rho_k}].$$

First, we observe that the vector $x(\rho_k)$ satisfying the condition

$$-(Gx^{k-1} + q)^T(x(\rho_k) - x^{k-1}) \geq \eta(x(\rho_k) - x^{k-1})^T G(x(\rho_k) - x^{k-1}) \tag{4.20}$$

can be found after a finite number of trials. Indeed, if we consider the first order optimality conditions of the subproblem (4.18) computed in $x = x^{k-1}$, we have

$$\begin{array}{rcl} (Gx^{k-1} + q)^T(x^{k-1} - x(\rho_k)) & \geq & \frac{1}{\rho_k}(x^{k-1} - x(\rho_k))^T D(x^{k-1} - x(\rho_k)) \\ & & -h(\rho_k)^T(x^{k-1} - x(\rho_k)). \end{array} \tag{4.21}$$

If m_k is the first nonnegative integer such that

$$\beta^{m_k}\overline{\rho}_k \leq \frac{\lambda_{min}(D)}{2\eta\lambda_{max}(G)}, \tag{4.22}$$

then the condition (4.20) is satisfied for $\rho_k = \beta^{m_k}\overline{\rho}_k$.

The inequality (4.22) states a very strong sufficient condition; from the viewpoint of the numerical effectiveness, it is advisable to define the projection parameter by a nonexpensive rule that gives values of ρ_k greater than that obtained by (4.22) and such that $m_k = 0$ for almost all k.
Now we prove the convergence of the AVPM for convex QP problems.

Theorem **3** *The sequence $\{x^k\}$, $k \geq 1$, generated by the AVPM is a well-defined sequence of vectors of K and it is convergent to a solution of (1.1) at least R–linearly.*

Proof. Since the solution of the subproblem (4.18) is unique and, at each iteration, the steps 2, 3 and 4 of the AVPM are repeated a finite number of times, x^k is a well-defined element of K.
Now the proof of the theorem runs as those of Theorems 1 and 2. First we prove that the following inequality holds:

$$f(x^{k-1}) - f(x^k) \geq \alpha_1 \|x^{k-1} - x^k\|^2 \tag{4.23}$$

where α_1 is a positive scalar.
Since x^k is such that the condition $-(Gx^{k-1}+q)^T(x^k - x^{k-1}) \geq \eta(x^k - x^{k-1})^T G(x^k - x^{k-1})$ holds, we can write

$$\begin{aligned} f(x^k) - f(x^{k-1}) &= (Gx^{k-1}+q)^T(x^k - x^{k-1}) \\ &\quad +\frac{1}{2}(x^k - x^{k-1})^T G(x^k - x^{k-1}) \\ &\leq (Gx^{k-1}+q)^T(x^k - x^{k-1}) \\ &\quad -\frac{1}{2\eta}(Gx^{k-1}+q)^T(x^k - x^{k-1}) \\ &\leq \left(1 - \frac{1}{2\eta}\right)(Gx^{k-1}+q)^T(x^k - x^{k-1}). \end{aligned} \tag{4.24}$$

From (4.21) with $x(\rho_k) = x^k$ and (4.19), it follows that

$$f(x^k) - f(x^{k-1}) \leq -\alpha_1 \|x^k - x^{k-1}\|^2 \tag{4.25}$$

where $\alpha_1 = \left(1 - \frac{1}{2\eta}\right)\frac{\lambda_{min}(D)}{\rho_M}$ and (4.23) holds. Then $\{f(x^k)\}$ is a sequence monotonically nonincreasing and, since $f(x)$ is bounded from below on K, it is convergent as $k \to \infty$ and $\lim_{k\to\infty} f(x^k) \geq f^*$. Consequently, the sequence $\{x^k - x^{k-1}\}$ converges to the null vector as $k \to \infty$. Now, we can prove that

$$\begin{aligned} \left\|x^{k-1} - [x^{k-1} - (Gx^{k-1}+q)]^+\right\| &= \Big\|x^{k-1} - [x^{k-1} - (Gx^{k-1}+q)]^+ + \\ &\quad -x^k + [x^k - \frac{D}{\rho_k}(x^k - x^{k-1}) - (Gx^{k-1} + q + h(\rho_k))]^+\Big\| \\ &\leq \alpha_2 \|x^k - x^{k-1}\| \end{aligned} \tag{4.26}$$

with $\alpha_2 = \left(2 + \frac{\lambda_{max}(D)+\lambda_{min}(D)}{\rho_m}\right)$. Since $\{\|x^k - x^{k-1}\|\}$ converges to 0 as $k \to \infty$, the sequence $\{\left\|x^{k-1} - [x^{k-1} - (Gx^{k-1}+q)]^+\right\|\}$ also converges to 0 as $k \to \infty$. As in Theorem 1, we can state that, for any k greater than a convenient $\overline{k}$, we have

$$\left\|x^{k-1} - [x^{k-1} - (Gx^{k-1}+q)]^+\right\| \leq \alpha_3 \tag{4.27}$$

$$\phi(x^{k-1}) \leq \alpha_4 \left\|x^{k-1} - [x^{k-1} - (Gx^{k-1}+q)]^+\right\| \tag{4.28}$$

and, finally,

$$\phi(x^{k-1}) \leq \alpha_5 \|x^k - x^{k-1}\| \tag{4.29}$$

where $\alpha_5 = \alpha_4 \cdot \alpha_2$; since $\{\|x^k - x^{k-1}\|\}$ converges to 0 as $k \to \infty$, the sequence $\{\phi(x^{k-1}\}$ converges to 0 as $k \to \infty$.
Let x^* the element of K^* closest to x^{k-1}, i.e. $\phi(x^{k-1}) = \|x^{k-1} - x^*\|$. From the first order optimality condition for the solution x^k of the subproblem (4.18) computed at $x = x^*$ and the Mean Value Theorem, it is possible to write

$$\left(\frac{D}{\rho_k}(x^k - x^{k-1}) + Gx^{k-1} + q + h(\rho_k)\right)^T (x^* - x^k) \geq 0,$$
$$f(x^k) - f(x^*) = (G\zeta^k + q)^T (x^k - x^*),$$

where ζ^k lies on the line segment joining x^k with x^*. Thus, combining the above relations, we have

$$\begin{aligned} f(x^k) \;&-\; f^* = f(x^k) - f(x^*) \\ &\leq \left(\lambda_{max}(G)\|\zeta^k - x^{k-1}\| + \|h(\rho_k)\| + \frac{\lambda_{max}(D)}{\rho_m}\|x^k - x^{k-1}\|\right) \|x^k - x^*\|. \end{aligned}$$

and, for $k > \overline{k}$, we can write

$$f(x^k) - f^* \leq \alpha_6 \|x^k - x^{k-1}\|^2$$

with $\alpha_6 = \left(\lambda_{max}(G)(\alpha_5 + 1) + \frac{\lambda_{max}(D) + \lambda_{min}(D)}{\rho_m}\right)(\alpha_5 + 1)$. Then, from (4.23), we have

$$f(x^k) - f^* \leq \frac{\alpha_6}{\alpha_1}(f(x^{k-1}) - f(x^k)), \qquad \text{for any } k > \overline{k}. \tag{4.30}$$

The R–linear convergence of $\{x^k\}$ follows as in Theorem 1.

4.3 Updating Rules for the Projection Parameter

The numerical behaviour of the variable projection methods depends on the choice of the sequence of projection parameters $\{\rho_k\}$. The only hypothesis required on $\{\rho_k\}$ for the convergence of these methods is that the sequence is bounded. Then, at the k-th iteration we can try to identify a "good" value for ρ_k on the basis of heuristic considerations from the last values of the quantities involved in the iterative scheme. In this way we can find some updating rules for varying the projection parameter in the VPM and in the AVPM. Of course, these rules can have different numerical effectiveness.
We may consider an updating rule for ρ_k to be numerically efficient when it enables an approximate solution to (1.1) to be obtained by a number of iterations of the VPM and the AVPM significantly lower than that of the classical projection and splitting methods. At the same time, the introduction of this rule must not increase the computational complexity of one iteration and must imply that the corrective step along a descent direction in the VPM and along a projection arc in the AVPM is present only in few iterations, i.e. $\theta_k = 1$ and, above all, $m_k = 0$ for almost all k.
In the following, we propose two updating rules.
The first is given by

$$\rho_k = \begin{cases} \rho_{k-1} & \text{for } \|Gd^{k-1}\|^2 \leq \psi \|d^{k-1}\|^2 \\ \frac{d^{k-1T} G d^{k-1}}{d^{k-1T} G D^{-1} G d^{k-1}} & \text{otherwise} \end{cases} \qquad k = 2, 3, \ldots \tag{4.31}$$

where $d^{k-1} = y^{k-1} - x^{k-2}$ in the VPM and $d^{k-1} = x^{k-1} - x^{k-2}$ in the AVPM. The positive scalar ψ is a prefixed small tolerance. The rule is a generalization of that proposed in [46] for strictly convex QP problems and it is deduced from the following considerations. We assume that the subproblem (4.2) is exactly solved, $h^{k-1} = 0$, for any $k \geq 1$ and $\{x^k\}$ converges to $x^* \in K^*$ as $k \to \infty$. By rearranging the first order conditions for problems (1.1) and (4.2) we obtain

$$\|x^* - y^k\|_D^2 \leq (x^* - x^{k-1})^T (D - \rho_k G)(x^* - y^k).$$

This last inequality holds in the VPM; for the AVPM we can substitute y^k with x^k. Now, by using the Cauchy–Schwarz inequality, we obtain

$$\|x^* - y^k\|_D^2 \leq \| (D - \rho_k G)(x^* - x^{k-1})\|_{D^{-1}} \|x^* - y^k\|_D$$

and, after dividing through by $\|x^* - y^k\|_D$, squaring and expanding, we have

$$\|x^* - y^k\|_D^2 \leq \tau_k \|x^* - x^{k-1}\|_D^2,$$

where

$$\begin{aligned}\tau_k &= 1 - \frac{\rho_k}{\|x^* - x^{k-1}\|_D^2}(2(x^* - x^{k-1})^T G(x^* - x^{k-1}) + \\ &\quad -\rho_k (x^* - x^{k-1})^T G D^{-1} G (x^* - x^{k-1})).\end{aligned}$$

If $(x^* - x^{k-1})^T G D^{-1} G (x^* - x^{k-1}) \neq 0$, the quantity τ_k is minimized for

$$\rho_k = \frac{(x^* - x^{k-1})^T G (x^* - x^{k-1})}{(x^* - x^{k-1})^T G D^{-1} G (x^* - x^{k-1})};$$

then, we can try to obtain a "good" value of ρ_k by using x^{k-2} in place of the solution x^*. For the VPM, we observe that $x^{k-1} - x^{k-2} = \theta_{k-1} d^{k-1}$. Since the vector Gd^{k-1} is already computed, the updating rule does not increase significantly the computational complexity of each iteration of the VPM and the AVPM.
In the context of the projection methods for variational inequalities, a similar value for the step size ρ_k has been deduced, again by heuristic arguments, in [14].
With the choice (4.31), in the VPM the sequence $\{\rho_k\}$ is bounded from below and above as follows:

$$\min\left(\rho_1, \frac{\lambda_{min}(D)}{\lambda_{max}(G)}\right) \leq \rho_k \leq \max\left(\rho_1, \frac{\lambda_{max}(G)\lambda_{max}(D)}{\psi}\right),$$

where the bounds are obtained by using the inequality $x^T G x \geq \frac{1}{\|G\|}\|Gx\|^2$.
In the AVPM, because of the adaptive procedure for determining ρ_k, the resulting sequence $\{\rho_k\}$ is bounded from below and above in the following way:

$$\min\left(\rho_1, \frac{\lambda_{min}(D)}{\lambda_{max}(G)}, \frac{\beta\lambda_{min}(D)}{2\eta\lambda_{max}(G)}\right) \leq \rho_k \leq \max\left(\rho_1, \frac{\lambda_{min}(D)}{2\eta\lambda_{max}(G)}, \frac{\lambda_{max}(G)\lambda_{max}(D)}{\psi}\right).$$

Another updating rule for ρ_k may be suggested by the inequality (4.20) when we assume $h(\rho_k) = 0$:

$$\rho_k = \frac{d^{kT} D d^k}{d^{kT} \eta G d^k} \quad \text{if } d^{kT} G d^k \neq 0,$$

where $d^k = x(\rho_k) - x^{k-1}$. Since $x(\rho_k)$ is not available, we use x^{k-2} instead of $x(\rho_k)$, obtaining the following rule for the projection parameter:

$$\rho_k = \begin{cases} \rho_{k-1} & \text{for } \|Gd^{k-1}\|^2 \leq \psi \|d^{k-1}\|^2 \\ \frac{d^{k-1T} D d^{k-1}}{d^{k-1} \eta G d^{k-1}} & \text{otherwise} \end{cases} \qquad k = 2, 3, \ldots \tag{4.32}$$

$d^{k-1} = x^{k-1} - x^{k-2}$ and ψ is a prefixed small tolerance. In the VPM we can use the rule (4.32) with $d^{k-1} = y^{k-1} - x^{k-2}$ and $\eta = 1$. Consequently, in the VPM the new sequence $\{\rho_k\}$ is bounded as in the previous case while, in the AVPM, the bounds for the new sequence $\{\rho_k\}$ are:

$$\min\left(\rho_1, \frac{\beta\lambda_{min}(D)}{2\eta\lambda_{max}(G)}\right) \leq \rho_k \leq \max\left(\rho_1, \frac{\lambda_{min}(D)}{2\eta\lambda_{max}(G)}, \frac{2\lambda_{max}(D)\lambda_{max}(G)}{\psi}\right).$$

Chapter 5

Solution of the Inner QP Subproblems

In all the projection-type and the splitting methods, as well as in the VPM and the AVPM, it is required to solve a sequence of strictly convex QP subproblems, having the following form

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}x^T\Delta x + q^{k-1T}x \\ \textit{subject to} & Cx = d, \qquad Ax \geq b, \end{array} \tag{5.1}$$

where $q^{k-1} = (G-\Delta)x^{k-1} + q$, $k = 1, 2, ...$, and Δ is an easily solvable matrix, for instance a diagonal or block diagonal matrix.
When the constraints have particular features, we can use specialized inner solvers. For example, for single constrained separable strictly convex QP problems with simple bounds on the variables see [36], [41] and references therein; for equality constrained strictly convex QP problems with simple bounds on the variables see the dual ascent methods in [25].
When the set of constraints does not present a particular structure, it is possible to formulate the subproblem (5.1) as a mixed LCP [25]. Indeed, by using the Karush–Kuhn–Tucker optimality conditions, we can derive the solution $\bar{x}$ of (5.1) in terms of its corresponding Lagrange multipliers $\bar{\lambda}$ and $\bar{\mu}$:

$$\bar{x} = \Delta^{-1}(-q^{k-1} + A^T\bar{\lambda} + C^T\bar{\mu})$$

where $\begin{pmatrix}\bar{\lambda}\\ \bar{\mu}\end{pmatrix}$ is the solution of the following mixed LCP:

$$\begin{pmatrix}u\\0\end{pmatrix} = \begin{pmatrix}z_1^{k-1}\\ z_2^{k-1}\end{pmatrix} + M\begin{pmatrix}\lambda\\ \mu\end{pmatrix} \tag{5.2}$$
$$u \geq 0, \quad \lambda \geq 0, \quad u^T\lambda = 0$$

where $z_1^{k-1} = -b - A\Delta^{-1}q^{k-1}$, $z_2^{k-1} = -d - C\Delta^{-1}q^{k-1}$ and the matrix

$$M = \begin{pmatrix}A\\C\end{pmatrix}\Delta^{-1}\begin{pmatrix}A^T & C^T\end{pmatrix} \tag{5.3}$$

is a symmetric positive semidefinite matrix of order $\nu = (m_i + m_e)$ with positive diagonal entries. The mixed LCP (5.2) is solvable because it arises from the strictly convex QP problem (4.18). In the case of equality constraints only, M is the symmetric positive definite matrix $CD^{-1}C^T$ and (5.2) is reduced to the following positive definite linear system:

$$M\mu = -z_2^{k-1}. \tag{5.4}$$

Thus, we can determine the solution of (5.1) by solving an equivalent problem whose size is equal to the number of constraints.
When this number is small, we can solve the problem (5.2) or (5.4) by direct methods.
In the case of large–scale sparse symmetric monotone LCPs, it appears convenient to use again a splitting iterative scheme, such as the classical Projected SOR scheme of Cryer [9] and the Projected Symmetric SOR scheme of Mangasarian [29]. In these schemes, starting from an arbitary vector $\begin{pmatrix}\lambda_0\\ \mu_0\end{pmatrix}$, with $\lambda_0 \geq 0$, we can generate the sequence of vectors $\begin{pmatrix}\lambda_i\\ \mu_i\end{pmatrix}$, by solving a sequence of mixed LCP subproblems, such as $(i = 1, 2, ...)$:

$$\begin{pmatrix}u\\ 0\end{pmatrix} = \begin{pmatrix}z_1^{k-1}\\ z_2^{k-1}\end{pmatrix} + Q\begin{pmatrix}\lambda_{i-1}\\ \mu_{i-1}\end{pmatrix} + P\begin{pmatrix}\lambda\\ \mu\end{pmatrix} \tag{5.5}$$
$$u \geq 0, \quad \lambda \geq 0, \quad u^T\lambda = 0$$

where (P, Q) is a P–regular splitting of M. The convergence of an iterative scheme as (5.5), can be derived from Theorem 1, by observing that the subproblem (5.2) is equivalent to the following convex QP problem:

$$\begin{array}{ll} \textit{minimize} & \frac{1}{2}\begin{pmatrix}\lambda\\ \mu\end{pmatrix}^T M\begin{pmatrix}\lambda\\ \mu\end{pmatrix} + \begin{pmatrix}z_1^{k-1}\\ z_2^{k-1}\end{pmatrix}^T\begin{pmatrix}\lambda\\ \mu\end{pmatrix} \\ \textit{subject to} & \lambda \geq 0 \end{array} \tag{5.6}$$

(see [11], [26]).
On parallel computers, we can easily achieve an efficient implementation of the projection type and splitting methods, as well as of the VPM and the AVPM, by distributing the matrix–vector products and the vector operations on the available processors and by using a parallel inner solver for the LCPs. A simple parallel solver for LCP is obtained by considering a splitting of the matrix M such that each iteration can be decomposed in independent processes. In [20] we show the effectiveness of the this approach in the case of a splitting method for QP problems combined with the Parallel SOR [31], the Parallel Gradient Projection SOR [32] and the Overlapping Parallel SOR [19] for the solution of the large inner LCPs.
When the inner subproblem (5.1) can be reformulated in the linear system (5.4), we can use as iterative solver the Preconditioned Conjugate Gradient Method, that is also well suited for implementation on parallel computers. The absence of a particular structure in the matrix M suggests to use as preconditioner the classical SSOR preconditioner [13] or the Arithmetic Mean preconditioner [15].
In the following we describe two splittings of M that give rise to different parallel solvers for the LCP.

5.1 Parallel solvers for symmetric LCPs

Consider the following feasible LCP:

$$\begin{array}{c} u = Mz + g \\ u \geq 0, \quad z \geq 0, \quad u^Tz = 0 \end{array} \tag{5.7}$$

where M is a symmetric positive semidefinite matrix of order ν, with positive diagonal entries. [1]
In order to introduce parallel iterative schemes for this problem, we partition the matrix M into p submatrices as follows:

$$M = \begin{pmatrix} \bar{M}_1 \\ \bar{M}_2 \\ \vdots \\ \bar{M}_p \end{pmatrix} \tag{5.8}$$

where $\bar{M}_i = (\,M_{i,1} \quad M_{i,2} \quad \dots \quad M_{i,p}\,)$, $M_{i,j} = \left(m_{k,l}^{(i,j)}\right)$, $i, j = 1, \dots, p$, are $\nu_i \times \nu_j$ matrices with $\displaystyle\sum_{i=1}^{p} \nu_i = \nu$ and $M_{j,i} = M_{i,j}^T$.
If we write $M_{i,i} = L_i + \Delta_i + L_i^T$, $i = 1, \dots, p$, where L_i is the strictly lower triangular part of $M_{i,i}$ and Δ_i its diagonal part, and we consider

$$\begin{array}{lcl} P & = & diag\left(E_1^{-1}\Delta_1 + L_1, \dots, E_p^{-1}\Delta_p + L_p\right) \\ Q & = & M - P \end{array} \tag{5.9}$$

where E_i, $i = 1, \dots, p$, is a diagonal matrix of order ν_i with nonzero entries $\omega_k^{(i)}$, $k = 1, \dots, \nu_i$, the splitting $M = P + Q$ generates the parallel iterative scheme proposed in [31], called Parallel SOR (PSOR) method. Starting from an arbitrary ν–vector $z^0 \geq 0$, the PSOR method generates a sequence of vectors $z^k = (\mathrm{w}_1^T, \dots, \mathrm{w}_p^T)^T$, by solving at each step p independent LCPs:

$$u_i = (E_i^{-1}\Delta_i + L_i)\mathrm{w}_i + (\Delta_i - E_i^{-1}\Delta_i + L_i^T)z_i^{k-1} + \sum_{\substack{j=1 \\ j \neq i}}^{p} M_{i,j} z_j^{k-1} + g_i$$

$$u_i \geq 0, \quad \mathrm{w}_i \geq 0, \quad u_i^T \mathrm{w}_i = 0 \quad i = 1, 2, \dots, p$$

where w_i, g_i and u_i are ν_i–vectors and $g = (g_1^T, \dots, g_p^T)^T$, $u = (u_1^T, \dots, u_p^T)^T$.
On a parallel computer with p processors, each of the p independent LCPs can be solved on a different processor.
If the following condition holds

$$0 < \omega_k^{(i)} < \frac{2}{1 + \dfrac{1}{m_{k,k}^{(i,i)}} \displaystyle\sum_{\substack{j=1 \\ j \neq i}}^{p} \sum_{l=1}^{\nu_j} |m_{k,l}^{(i,j)}|} \tag{5.10}$$

($k = 1, \dots, \nu_i$, $i = 1, \dots, p$), for any $z^0 \geq 0$, the sequence $\{z^k\}$ is convergent to a solution of (5.7) as $k \to \infty$. Indeed, in this case, it is immediate to prove that $P+Q$ is a P–regular splitting of the symmetric positive semidefinite matrix M.
We point out that for $p = 1$ and $\omega_k^1 \equiv \omega$ for any k, we have $0 < \omega < 2$ and PSOR is equal to the Cryer's method [9]. For $p > 1$, the convergence condition (5.10) imposes that the relaxation factors ω_k^i must be in intervals whose upper bounds are less than 2 and this can lead to a slow convergence.
For a given matrix M without a quasi–band structure, generally the upper bound for ω_k^i decreases as p increases and, consequently, the number of iterations of the PSOR method may increases as the number p of available processors increases.

[1]For simplicity, we consider the standard form of the LCP; it is easy to derive the following considerations for the special case of a mixed LCP.

To overcome the disadvantage of the PSOR method, Mangasarian and De Leone propose in [32] a gradient projection–SOR (GPSOR) algorithm that, when appropriately parallelized, converges with the relaxation factors in $(0, 2)$.
In order to summarize this approach we consider the following convex QP problem

$$\begin{array}{ll} \textit{minimize} & \phi(z) = \frac{1}{2} z^T M z + g^T z \\ \textit{subject to} & z \geq 0 \end{array} \tag{5.11}$$

and note that the LCP (5.7) express the optimality conditions for (5.11).
The GPSOR method generated by the splitting $M = P + Q$ can be expressed as follows:

1. Let z^0 be an arbitrary ν-vector such that $z^0 \geq 0$; $k \leftarrow 1$.
2. Define the direction $d^k = p(z)^k - z^{k-1}$, where $p(z)^k$ is the solution of the following LCP:
$$u = g + Qz^{k-1} + Pp(z)$$
$$u \geq 0, \quad p(z) \geq 0, \quad u^T p(z) = 0$$
3. Terminate if $d^k = 0$; otherwise $z^k = z^{k-1} + \lambda_k d^{(k)}$, where
$$\phi(z^{k-1} + \lambda_k d^k) = \min_{\lambda} \{\phi(z^{k-1} + \lambda d^k),\ z^{k-1} + \lambda d^k \geq 0\}$$
4. Terminate if some appropriate stopping rule is satisfied; otherwise $k \leftarrow k + 1$ and go to step 2.

For a symmetric matrix M, the convergence condition for the GPSOR method is the matrix P positive definite [32]. When we consider P as in (5.9), we have the parallel version of GPSOR method (PGPSOR). In this case, if M is symmetric positive semidefinite with positive diagonal entries, the method is convergent for $0 < \omega_k^i < 2$ $(k = 1, \ldots, \nu_i,\ i = 1, \ldots, p)$ since

$$P + P^T = diag((2E_1^{-1} - I)\Delta_1, \ldots, (2E_p^{-1} - I)\Delta_p) + diag(M_{1,1}, \ldots, M_{p,p})$$

and then P is positive definite.
On a multiprocessor system the matrix–vector product $s^k = Md^k$ used to compute λ_k, can be obtained in parallel. Then on the i–th processor the computational complexity of one iteration is proportionally twice the number of nonzero entries of the submatrix $\bar{M}_i$. Furthermore, before to compute λ_k, each processor must send its computed part of the vector $p(z)^k$ to the other processors and must receive the other parts from the other processors. The same operation must be repeated for the vector s^k. This global communication operation is known as multinode broadcast [5]. In this case the operation must be synchronous (i.e. all processors must expect the completion of the operation before to proceed). In order to balance the workload among the processors, we set $\nu_i = \nu/p$, for any i (we assume that p is a divisor of ν). Thus the time to execute one PGPSOR iteration on p processors can be expressed as

$$t_{PGPSOR} = t_{computation} + t_{communication} = 2 * (O(\eta\nu/p)t_{fl} + t_{smb}(\nu/p, p))$$

where t_{fl} is the time to perform a floating point operation, $\eta\nu/p$ $(\eta << \nu)$ is the maximum number of nonzero entries of the submatrices $\bar{M}_i$ $(i = 1, \ldots, p)$ and t_{smb} is the time of a synchronous multinode broadcast among p processors, each sending ν/p data to the

others. We observe that t_{PGPSOR} is twice the time to execute one iteration of the PSOR method.
Another way to improve the PSOR method is to consider a different splitting of the matrix M, by partitioning the matrix M into p submatrices as in (5.8) where

$$\bar{M}_1 = \begin{pmatrix} M_{1,1} & M_{1,2} & \dots & M_{1,2p-1} \end{pmatrix}$$

$$\bar{M}_l = \begin{pmatrix} M_{2l-2,1} & M_{2l-2,2} & \dots & M_{2l-2,2p-1} \\ M_{2l-1,1} & M_{2l-1,2} & \dots & M_{2l-1,2p-1} \end{pmatrix} \qquad l = 2, \dots, p-1$$

$$\bar{M}_p = \begin{pmatrix} M_{2p-2,1} & M_{2p-2,2} & \dots & M_{2p-2,2p-1} \\ M_{2p-1,1} & M_{2p-1,2} & \dots & M_{2p-1,2p-1} \end{pmatrix}$$

and $M_{i,j} = \left(m_{k,l}^{(i,j)}\right)$, $i,j = 1,\dots,2p-1$, are $\nu_i \times \nu_j$ matrices with $\sum_{i=1}^{2p-1} \nu_i = \nu$ and $M_{j,i} = M_{i,j}^T$.
If we write $M_{i,i} = L_i + \Delta_i + L_i^T$, $i = 1,\dots,2p-1$ and we consider $P = (P_{i,j})$, $i,j = 1,\dots,2p-1$, with

$$P_{i,j} = \begin{cases} E_i^{-1}\Delta_i + L_i & \text{for} \quad i = j, \quad i = 1,\dots,2p-1 \\ M_{i,j} & \text{for} \quad i = 1,3,\dots,2p-1, \quad j = 2,4,\dots,2p-2, \\ 0 & \text{otherwise} \end{cases}$$

where E_i, $i = 1,\dots,2p-1$, is a diagonal matrix of order ν_i with nonzero entries ω_k^i, $k = 1,\dots,\nu_i$, and $Q = M - P$, then the splitting $M = P + Q$ generates an iterative scheme, called Overlapping Parallel SOR (OPSOR) method. As for the PSOR method, starting from an arbitrary ν-vector z^0, such that $z^0 \geq 0$, the OPSOR method generates a sequence of vectors $z^k = (\mathrm{w}_1^T, \mathrm{v}_2^T, \mathrm{w}_3^T, \dots, \mathrm{v}_{2p-2}^T, \mathrm{w}_{2p-1}^T)^T$ by solving the following two sets of LCPs:

1. compute the solutions v_i, $i = 2,4,\dots,2p-2$, of the $(p-1)$ LCPs

$$u_i = (E_i^{-1}\Delta_i + L_i)\mathrm{v}_i + (\Delta_i - E_i^{-1}\Delta_i + L_i^T)z_i^{k-1} + \sum_{\substack{j=1 \\ j\neq i}}^{2p-1} M_{i,j} z_j^{k-1} + g_i$$

$$u_i \geq 0, \quad \mathrm{v}_i \geq 0, \quad u_i^T \mathrm{v}_i = 0$$

2. compute the solutions w_i, $i = 1,3,\dots,2p-1$, of the p LCPs

$$\begin{aligned} u_i \quad = \quad & (E_i^{-1}\Delta_i + L_i)\mathrm{w}_i + (\Delta_i - E_i^{-1}\Delta_i + L_i^T)z_i^{k-1} + \\ & + \sum_{j \text{ even}} M_{i,j}\mathrm{v}_j + \sum_{\substack{j \text{ odd} \\ j\neq i}} M_{i,j} z_j^{(k-1)} + g_i \\ & u_i \geq 0, \quad \mathrm{w}_i \geq 0, \quad u_i^T \mathrm{w}_i = 0 \end{aligned}$$

Here v_i, w_i, g_i and u_i are ν_i–vectors and $g = (g_1^T, \dots, g_{2p-1}^T)^T$, $u = (u_1^T, \dots, u_{2p-1}^T)^T$.
As for PSOR method, if the LCP (5.7) is feasible and if the following conditions hold $(k = 1,\dots,\nu_i)$

$$
(5.12) \qquad
\begin{array}{ll}
0 < \omega_k^i < \dfrac{2}{1 + \dfrac{1}{m_{k,k}^{(i,i)}} \displaystyle\sum_{\substack{j\text{even}\\ j\neq i}} \sum_{l=1}^{\nu_j} |m_{k,l}^{(i,j)}|} & i \text{ even} \\[2em]
0 < \omega_k^i < \dfrac{2}{1 + \dfrac{1}{m_{k,k}^{(i,i)}} \displaystyle\sum_{\substack{j\text{odd}\\ j\neq i}} \sum_{l=1}^{\nu_j} |m_{k,l}^{(i,j)}|} & i \text{ odd}
\end{array}
$$

then, for any z^0 the sequence $\{z^k\}$ generated by OPSOR method converges to a solution of (5.7) as $k \to \infty$.
Indeed if conditions (5.12) hold, $R = (R_{i,j}) = P^T - Q$

$$
R_{i,j} = \begin{cases} \left(2E_i^{-1} - I\right)\Delta_i & \text{for } \quad i = j, \quad i = 1, \ldots, 2p-1 \\ -M_{i,j} & \text{for } \quad i \neq j, \quad i,j \text{ even or } i,j \text{ odd} \\ 0 & \text{otherwise} \end{cases}
$$

is a symmetric strictly diagonally dominant matrix with positive diagonal entries then is a symmetric positive definite matrix. Since $P = \frac{1}{2}(M + P - Q)$, P is a positive definite matrix and then it is nonsingular and $P + Q$ is a P–regular splitting of M.
For a given sparse matrix M without a quasi–band structure, also for the OPSOR method the upper bound for ω_k^i generally decreases as p increases. But this upper bound does not depend on all the entries of the corresponding row of M. Then (as showed experimentally in [19]) given p, the number of iterations required by OPSOR method for solving the LCP (5.7) is less than that required by PSOR method.
On a parallel computer with p processors, the $(p-1)$ independent LCPs at the step 1 can be executed simultaneously on different processors and, analogously, the p independent LCPs at the step 2. Then, the complexity of one iteration of the OPSOR method on the l-th processor is proportional to the number of nonzero entries of the l-th submatrix $\bar{M}_l$ of M.
After steps 1 and 2, by two synchronuos multinode broadcast operations, each active processor sends the computed part of the vector z^k to the other processors and each processor receives the parts of z^k computed by the other processors.
In order to balance the workload among the processors, in practice we choose a value ν_{over} for ν_i with i even and we set $\nu_i = \bar{\nu} = \frac{\nu-(p-1)\nu_{over}}{p}$ for i odd.
Thus the time to execute one OPSOR iteration on p processors can be expressed as

$$
\begin{aligned}
t_{OPSOR} &= t_{computation} + t_{communication} = \\
&= O(\alpha(\nu_{over} + \bar{\nu}))t_{fl} + t_{smb}(\nu_{over}, p) + t_{smb}(\bar{\nu}, p)
\end{aligned}
$$

where $\alpha(\nu_{over} + \bar{\nu})$ $(\alpha << \nu)$ is the maximum number of nonzero entries of the p submatrices of M. It is evident that the time to execute one iteration of OPSOR method is greater than that of the PSOR method.

Chapter 6

Computational Experiments

In this Chapter we report the results of a set of numerical experiments that show the computational behaviour of the methods described in the previous chapters. The experiments are carried out by a set of programs written in Fortran 77 on a Digital Alpha Workstation 500/333 Mhz (Tables 1–5) and on a Digital Personal Workstation 500au (Tables 6–10), using the double precision (macheps= $2.22 \cdot 10^{-16}$). The results reported in Tables 11–13 are carried out on a Cray MPP T3E/128–128 at the C.I.N.E.C.A. Supercomputing Center [1].
The most part of the considered test problems are randomly generated with assigned features by following a technique similar to that introduced in [50], but using Givens rotations instead of Householder elementary trasformations for obtaining a prespecified level of sparsity [44]. In particular, in these test problems, we prefix the sizes n, m_e, m_i, a solution x^* and the corresponding Lagrange multipliers, the number *nac* of inequality constraints that are active in the solution x^*, the level of sparsity (denoted by spars$(\cdot)$), the spectral condition number (denoted by $K(\cdot)$), the rank, the euclidean norm and the distribution of singular values of the matrices G and $\begin{pmatrix} A \\ C \end{pmatrix}$.
The values considered for several features of the test problems (e.g. size, condition number, level of sparsity) reflect those of the problems arising in many practical applications. Furthermore, even if these experiments concern test problems without structure, it is well known that the splitting and projection–type methods are suited to exploit the structures of the Hessian matrix or the constraint matrices that often appear in the real problems.
In all the experiments we use the following stopping rule:

$$\frac{\|x^{k+1} - x^k\|}{\|x^{k+1}\|} \le tol \tag{6.1}$$

where $tol = 10^{-6}$ in Table 1–5, 7–10, $tol = 10^{-12}$ in Table 6 and $tol = 10^{-9}$ in Tables 11–13. In the tables we denote by *it* and *time* the number of iterations and the time in seconds to obtain the solution and by er_x the relative error $\frac{\|x^* - x^{it}\|}{\|x^*\|}$.
The results reported in the following tables are related to the following subjects:

[1]The Cray T3E is a MIMD system, scalable up to 2048 processing elements (PEs), connected by a three–dimensional torus network, with a bandwidth of 480 Mbytes/sec. on each direction. Each PE has a *local memory*, but can access to the local memory of all other PEs (*remote memory*), making the *physically distributed* memory *globally addressable*,
The Cray T3E at C.I.N.E.C.A. Supercomputing Center has 128 PEs; each PE has 128 Mbytes of local memory and a peak performance of 600 Mflops.

- numerical behaviour of the projection type methods for medium–scale and large–scale QP problems;
- comparison between the VPM and the active set method for large–scale QP problems;
- comparison among the scaled gradient projection methods and the variable projection–type methods;
- numerical behaviour of parallel inner solvers in the projection–type methods.

Table 1. Behaviour of the PM for different values of ρ

$n = 1000$ $m_e = 600$ $m_i = 0$; spars$(G) = 98\%$ $\|G\|_2 = 1$; spars$(C) = 99\%$ $\|C\|_2 = 10$ $K(C) = 10^2$; sufficient convergence condition: $\rho < 2$				
Projection Method				
$K(G)$	ρ	*it*	*time*	er_x
10^2	1.43	405	14.2	2.7e-5
	1.67	347	12.3	2.6e-5
	1.94	297	11.0	2.7e-5
	1.98	658	23.3	1.2e-7
	1.995	2742	94.3	1.2e-7
10^3	1.43	2455	84.2	2.7e-4
	1.67	2104	72.4	2.7e-4
	1.94	1806	61.8	2.7e-4
	1.98	1771	60.2	2.7e-4
	1.995	2741	93.7	3.8e-5
10^4	1.43	3800	129.1	5.9e-4
	1.67	3246	111.0	5.9e-4
	1.94	2796	98.9	5.9e-4
	1.98	2741	93.9	5.9e-4
	1.995	2721	93.4	5.9e-4
10^5	1.43	7215	246.3	3.2e-3
	1.67	6178	211.7	3.2e-3
	1.94	5319	181.6	3.2e-3
	1.98	5210	177.5	3.2e-3
	1.995	5171	176.2	3.2e-3

6.1 Numerical behaviour of the projection type methods for medium–scale and large–scale QP problems

The aim of the first set of experiments is to evaluate the numerical features of the considered methods, removing the possible dependencies on the inner iterative solver. Thus, we use test problems of medium–size ($n = 1000$) with equality constraints only ($m_e = 600$), so that the inner subproblems can be solved by LAPACK routines. In this

first set of experiments, the eigenvalues of G have a uniform distribution, the sparsity levels of G and C are 98% and 99% respectively, $\|G\|_2 = 1$, $\|C\|_2 = 10$ and $K(C) = 10^2$. The results of these tests are reported in Tables 1 and 2.
With regard to the SM, a P–regular splitting of G is obtained by taking

$$D = \Omega \text{diag}(g_{11}, g_{22}, ..., g_{nn}),$$

where Ω is a diagonal matrix with entries ω_i, $i = 1, ..., n$, satisfying the condition $\omega_i > \max\left(1, \frac{\sum_j |g_{ij}|}{2g_{ii}}\right)$; consequently, $2D - G$ is a strictly diagonally dominant matrix with positive entries and, then, it is positive definite. In the PM, we choose $D = I$. For the MPM, we use $P = I + G$ and we report the results obtained with the empirical optimal value of parameter θ ($\theta = 0.8$). An "a priori" scaling of G and q is performed using the technique suggested in [49]. For the VPM, we use $D = I$ and the rule (4.31) for updating the parameter ρ_k at each iteration.
Table 1 shows the behaviour of the PM with respect to different values of the parameter ρ. For each prefixed value of the condition number $K(G)$ of G, we try to individuate the value of ρ that satisfies the sufficient convergence condition $\rho < 2$ and gives the better performance. Nevertheless, in some cases, it is possible to find values of ρ that do not satisfy the sufficient convergence condition and produce a lower number of iterations; for example, when $K(G) = 10^4$ and $\rho = 3.92$, we have $it = 1463$, $time = 51.1$, $er_x = 5.1e-4$ and when $K(G) = 10^5$ and $\rho = 3.98$, we obtain $it = 3199$, $time = 109.8$, $er_x = 2.5e-3$.

Table 2. Comparison among the PM, the SM, the MPM and the VPM.

	PM			SM			MPM			VPM		
$K(G)$	*it*	*time*	er_x	*it*	*time*	er_x	*it*	*time*	er_x	*it*	*time*	er_x
10^2	297	11.0	2.7e-5	310	12.2	2.5e-5	167	7.9	2.6e-5	58	3.3	1.1e-5
10^3	1771	60.2	2.7e-4	1993	73.4	2.5e-4	879	42.4	2.7e-4	139	7.0	2.0e-4
10^4	2721	93.4	5.9e-4	2454	90.3	6.0e-4	1507	67.5	4.3e-4	155	7.7	5.9e-4
10^5	5171	176.2	3.2e-3	7519	266.3	2.2e-3	2563	112.6	1.3e-3	181	8.8	2.4e-4

In Table 2 we can observe how the number of iterations of all the methods changes when $K(G)$ increases; for the PM we report the better results obtained with ρ satisfying the sufficient convergence condition. The MPM and VPM are more efficient than the classical SM and PM; in particular, the VPM requires in all cases a very small number of iterations and seems to be weakly dependent on the condition number of G. Furthermore, the results obtained by the VPM do not require an initial scaling of G and q, while the efficiency of the MPM is strongly affected by the use of this scaling (for example, for the test problems of Table 2, if we do not use the scaling suggested in [49], it results: $it = 311$ for $K(G) = 10^2$, $it = 1853$ for $K(G) = 10^3$, $it = 2864$ for $K(G) = 10^4$ and $it = 5437$ for $K(G) = 10^5$). Finally, the rate of convergence of the methods also depends on the distribution of the eigenvalues of G. For example, in the case of the SM, when $K(G) = 10^4$ and we have some hundreds of eigenvalues of G close to the maximum eigenvalue, the number of iterations is 63; on the contrary, if some hundreds of eigenvalues are close to the minimum eigenvalue, $it = 4222$.
When we use a direct solver for solving the inner subproblems, all the considered methods are not affected by the increase of the condition number of the constraint matrix.
On the contrary, the choice of an iterative scheme as inner solver is crucial for the effectiveness of the methods. Table 3 shows the numerical behaviour of the MPM and the VPM when we use the projected SOR method as inner solver [9]. In this case, we choose an empirical optimal value for the SOR relaxation parameter ($\omega = 1.5$) and an inner progressive termination rule depending on the quality of the previous outer

iterate. We consider equality and inequality constrained test problems (nac is the number of inequality constraints that are active in the solution x^*) with a constant value for $K(G)$ and different values for $K((\, C^T \;\, A^T \,)^T)$. We point out that the number of outer iterations is weakly dependent on $K((\, C^T \;\, A^T \,)^T)$, while the number of inner iterations (reported in brackets) increases markedly when $K((\, C^T \;\, A^T \,)^T)$ increases, and this affects the performance of the methods.

Table 3. Behaviour of the MPM and the VPM with an iterative inner solver

$n = 2000$ $\quad m_e = 400$ $\quad m_i = 600$ $\quad nac = 400$ $K(G) = 10^4$ $\quad$ spars$(G) = 99\%$ $\quad$ spars$((\, C^T \;\, A^T \,)^T) = 99.6\%$						
	MPM (SOR)			VPM (SOR)		
$K((\, C^T \;\, A^T \,)^T)$	*it*	*time*	er_x	*it*	*time*	er_x
10	361 (1740)	7.6	3.9e-5	32 (649)	1.6	1.2e-5
100	425 (19898)	28.7	3.9e-5	37 (11931)	13.9	4.1e-6
300	491 (97171)	110.5	3.9e-5	38 (24508)	26.6	6.3e-6

It is interesting to observe that the behaviour of the projection–type methods depends on the numerical performance of the inner iterative solver. In Table 4 we consider test problems with equality constraints only and we report a comparison between two different inner solvers for the VPM: the projected SOR method [9] and the PCG method with the Arithmetic Mean preconditioner [15]. In this case $K(G)$ is fixed and we change $K(C)$; furthermore, in order to avoid the introduction of scaling techniques on the inner solvers, the constraint matrices are generated so that $\|C\|_2 = 1$.

We point out that the number of outer iterations is about the same in all the cases, while the number of inner iterations increases as $K(C)$ increases. Nevertheless, the efficiency of the VPM is preserved if we use the PCG as inner solver.

In Table 5 we show the results obtained by the SM, the MPM and the VPM for well conditioned test problems with size $n = 8000$. We use as inner solver the PCG method for the equality constrained problem and the Projected SOR method for the equality and inequality constrained problem.

We may observe that, also for these cases, the VPM achieves the best performance; furthermore, since the VPM is weakly dependent on the condition number of G (see Table 2) it appears promising also for the solution of not so well conditioned large–scale problems.

Table 4. Behaviour of VPM with different iterative inner solvers

$n = 1000$ $\quad m_e = 600$ $\quad m_i = 0$ spars$(G) = 98\%$ $\quad \|G\|_2 = 1$ $\quad K(G) = 10^3$ spars$(C) = 99\%$ $\quad \|C\|_2 = 1$						
	VPM (SOR)			VPM (PCG)		
$K(C)$	*it*	*time*	er_x	*it*	*time*	er_x
10	29 (569)	1.6	2.1e-5	30 (485)	2.4	4.4e-6
30	31 (2836)	5.4	2.2e-6	31 (1082)	4.8	4.5e-7
50	30 (4323)	6.1	2.2e-5	29 (1355)	4.4	1.5e-6
100	27 (8480)	11.4	1.5e-5	29 (1881)	5.9	1.8e-6
300	35 (21973)	28.0	9.2e-6	29 (2385)	7.1	3.5e-6

Table 5. Well conditioned large–scale QP problems

$n = 8000$

$\text{spars}(G) = 99.8\%$ $\quad \|G\|_2 = 40$ $\quad K(G) = 30$

$\text{spars}\begin{pmatrix} A \\ C \end{pmatrix} = 99.9\%$ $\quad \left\|\begin{pmatrix} A \\ C \end{pmatrix}\right\|_2 = 1$ $\quad K\begin{pmatrix} A \\ C \end{pmatrix} = 10$

	SM			MPM			VPM		
	it	*time*	er_x	*it*	*time*	er_x	*it*	*time*	er_x
$m_e = 3000$ $m_i = 0$	222 (2137)	128.7	3.5e-6	151 (1010)	116.2	2.4e-6	46 (285)	43.6	1.9e-6
$m_e = 2000$ $m_i = 3000$ $nac = 2000$	235 (1532)	169.9	3.5e-6	144 (814)	136.3	3.7e-6	48 (481)	71.9	2.1e-6

It is important to remark that the test problems considered in Tables 1–5 are randomly generated. When the underlying application suggests a convenient P–regular splitting for the Hessian matrix G, the efficiency of the SM may increase (see Chapter 7).

6.2 Comparison among the VPM and the active set method for large–scale QP problems

In Table 6 we consider a set of large and very sparse QP test problems randomly generated with high condition numbers for the Hessian matrix and the constraint matrix. The solution of this kind of problem by splitting and projection methods requires an excessive number of iterations, while the VPM gives an approximation of the solution in a reasonable time. Furthermore, the results of Table 6 allow us to compare the numerical effectiveness of the VPM with the active–set method implemented in the routine E04NKF of the NAG library [34]. This routine is designed for sparse QP problems and it based on parts of the SNOPT and MINOS packages.

Table 6. Large–scale QP problems

$n = 5000$

$\|G\| = 1$ $\quad K(G) = 10^4$ $\quad \text{spars}(G) = 99.90\%$

$\|\,(C^T\ A^T\,)^T\,\| = 1$ $\quad K((\,C^T\ A^T\,)^T) = 10^3$ $\quad \text{spars}((\,C^T\ A^T\,)^T) = 99.95\%$

			VPM			E04NKF	
m_e	m_i	nac	*it*	*time*	er_x	*time*	er_x
500	0	0	208 (1861)	3.4	6.6(-11)	46.1	5.3(-15)
4900	0	0	206 (208)	10.8	2.2(-11)	11.0	1.0(-14)
4700	1800	100	166 (957)	29.3	5.0(-11)	16.0	4.8(-15)
4000	2500	800	167 (1308)	34.9	2.3(-11)	44.4	4.0(-15)
3000	3500	1800	136 (1182)	32.4	1.9(-11)	134.0	1.1(-14)
3950	550	50	150 (871)	8.4	4.8(-11)	18.4	7.7(-15)
3000	1500	1000	158 (1344)	9.5	2.1(-11)	151.8	6.4(-15)
2500	2000	1500	176 (1722)	10.7	3.1(-11)	246.5	6.1(-15)

In order to improve the coherence in the accuracy of the two approaches, the stopping rule (6.1) for the VPM works with $tol = 10^{-12}$; in this way, starting from the same infeasible point, we have $|f(x^{it}) - f(x^*)|/|f(x^*)| \leq 10^{-15}$ for both methods. In all the following tests, the inner solver used by the VPM is the PCG method for equality constrained problems and the projected SOR method for equality and inequality constrained problems. The results of Table 6 show that, for the QP problems with only equality constraints, the behaviour of the two methods is reversed: when $m_e \ll n$ the VPM appears very convenient, while this is the worst case for the NAG routine. In the case of QP problems with equality and inequality constraints, the routine E04NKF, since it is based on an active–set strategy, benefits by a small value of *nac*. On the other hand, for a given number of constraints $(m_e + m_i)$, the behaviour of the VPM appears weakly dependent on *nac*.

6.3 Comparison among the scaled gradient projection methods and the variable projection–type methods

Tables 7 and 8 show the numerical behaviour of the VPM and the AVPM with respect to the class of the gradient projection methods. In particular, in this comparison, we consider the following schemes:

– the projection method with a limited minimization rule [4, p. 205]. This scheme is equal to the VPM with $D = I$ and $\rho_k = \rho$ for any k; it is denoted by FPM in Table 7.

– the projection method with an Armijo rule along a projection arc [4, p. 206]; for the choice of the projection parameter we can use the following strategy: given a positive integer γ and a constant value $\rho > 0$, at the k–th iteration with k such that $mod(k-1, \gamma) = 0$, we start the search procedure with the value ρ and, for the next $(\gamma - 1)$ iterations we use as projection parameter the value obtained by the search procedure in the previous iteration. In practice, the scheme is equal to the AVPM with $D = I$ and $\overline{\rho}_k = \rho$ when $mod(k-1, \gamma) = 0$ and $\overline{\rho}_{k+i+1} = \rho_{k+i}$ for $i = 0, ..., \gamma - 2$. When $\gamma = 1$, $\overline{\rho}_k = \rho$ for any $k \geq 1$. The above scheme is denoted by AFPM-γ in Tables 7 and 9.

For all methods, the inner QP subproblems are solved by the projected SOR method of Cryer.
With $tol = 10^{-6}$ in the stopping rule (6.1), we obtain that $|f(x^{it}) - f(x^*)|/|f(x^*)| \leq 10^{-11}$ for all methods.
In Tables 7 and 8 we report the results of the FPM, the AFPM-1, the AFPM-10, the VPM and the AVPM for the same test problems; these last two methods are combined with the rules (4.31) and (4.32) for ρ_k. In the VPM and in the AVPM we put $D = I$.

Furthermore, we denote by n_c the number of correction steps performed after the line search in the FPM and the VPM. In the AFPM-γ and in the AVPM, n_c is the number $\sum_{k=1}^{it} m_k$ of the additional projection steps performed in the search procedure of the projection parameter. In these last methods, $\beta = 0.5$ and $\eta = 0.9$.
From Tables 7 and 8, we can make the following remarks:

– in the gradient projection methods (Table 7), as ρ increases, the number *it* of outer iterations goes down to a minimum value and then begins to increase again. The variation of *it* is more consistent in the AFPM-γ. Nevertheless, while the line search procedure and the corrective step of the FPM are nonexpensive, the additional projection steps in the AFPM-γ affect the computational complexity considerably. In particular, in AFPM-1 the adaptive search procedure is always performed for large

Table 7

$n = 5000$

spars$(G) = 99.8\%$ $\|G\| = 1$ $K(G) = 100$

spars$((A^T\ C^T)^T) = 99.9\%$ $\|(A^T\ C^T)^T\| = 1$ $K((A^T\ C^T)^T) = 10$

	FPM			AFPM-10			AFPM-1		
ρ	*it*	n_c	*time*	*it*	n_c	*time*	*it*	n_c	*time*
$m_e = 1000$ $m_i = 1000$ $nac = 500$ rank$(G) = 4900$									
1.4	557 (1747)	1	35.9	557 (1747)	0	34.7	557 (1747)	0	35.3
2	403 (1277)	290	26.0	401 (1290)	1	25.5	398 (1274)	2	25.7
3.3	398 (1272)	398	26.0	333 (1205)	36	23.7	329 (1789)	225	35.1
7	399 (1312)	399	25.4	241 (1022)	60	19.2	249 (2157)	405	39.9
20	393 (1448)	393	25.3	163 (944)	75	15.9	220 (3006)	683	54.7
100	344 (2692)	340	24.7	92 (882)	65	11.4	239 (5978)	1340	95.4
200	622 (5123)	620	44.6	122 (1214)	96	15.1	330 (9692)	2177	152.0
1000	1866 (20475)	1864	138.5	146 (2071)	142	20.5	265 (13231)	2281	158.2
$m_e = 2000$ $m_i = 2000$ $nac = 1000$ rank$(G) = 4900$									
1.4	517 (1617)	1	47.4	517 (1616)	0	47.5	517 (1616)	0	47.8
2	374 (1185)	72	36.3	373 (1204)	1	36.4	370 (1189)	2	36.7
3.3	347 (1116)	347	34.4	321 (1602)	40	38.2	339 (2525)	258	59.2
7	333 (1154)	324	33.4	242 (1651)	64	35.1	238 (3380)	406	66.5
20	305 (1797)	258	35.8	140 (1617)	61	28.5	203 (2838)	633	73.9
100	909 (5487)	885	96.4	108 (1575)	72	26.9	247 (6140)	1385	144.4
200	1951 (11850)	1928	201.8	250 (3754)	193	57.2	308 (9138)	2033	206.2
$m_e = 3500$ $m_i = 1000$ $nac = 500$ rank$(G) = 4900$									
1.4	481 (1513)	3	54.6	482 (1523)	1	53.7	481 (1520)	1	55.6
2	356 (1126)	340	42.1	346 (1120)	1	41.0	343 (1104)	2	42.8
3.3	356 (1134)	356	42.3	289 (1050)	32	38.5	287 (1568)	198	56.0
7	197 (700)	184	27.1	206 (940)	59	34.2	239 (2139)	401	69.8
20	206 (1080)	171	31.6	145 (1129)	66	32.4	192 (2785)	594	84.8
100	467 (2927)	456	65.7	103 (1136)	71	30.2	228 (6403)	1281	168.3
200	1606 (9008)	1606	196.9	123 (1477)	97	36.2	292 (9837)	1928	248.1

values of ρ and it requires a considerable number of projection steps. Consequently, the AFPM-10 is generally more efficient than the FPM (because of the lower value of *it*) and it is more efficient than the AFPM-1 (because of the lower value of n_c). The highest performance of AFPM-10 is obtained for the minimum value of $(it + n_c)$.

- The VPM and the AVPM combined with the updating rule (4.31) have a similar numerical behaviour (Table 8). They are more efficient than the gradient projection methods: the number of outer iterations is lower and the corrective steps of the VPM or the additional projection steps of the AVPM arise in few iterations. Furthermore, the VPM does not require any prefixed scalar parameter.

- The updating rule (4.32) derived from heuristic considerations about the AVPM is nonconvenient when it is combined with the VPM. In the case of the AVPM as well, rule (4.32) appears less efficient than rule (4.31). Indeed, when we use rule (4.32), the total number of the iterations of the inner solver increases.

The above considerations about the numerical behaviour of the AFPM-10 [2], the VPM and the AVPM combined with rule (4.31) are confirmed in Table 9, where we consider some test problems with a large number of null eigenvalues in the Hessian matrix G. Here, Z denotes the $n \times (n - (m_e + nac))$ matrix having as columns an orthonormal basis of the null space of the matrix of the equality constraints and of the inequality

[2] In Table 9, for the AFPM-10, we report the results obtained with an empirical optimal value of ρ.

Table 8

$n = 5000$ spars$(G) = 99.8\%$ $\|G\| = 1$ $K(G) = 100$ spars$((A^T\ C^T)^T) = 99.9\%$ $\|(A^T\ C^T)^T\| = 1$ $K((A^T\ C^T)^T) = 10$											
AVPM – rule (4.31)			AVPM – rule (4.32)			VPM – rule (4.31)			VPM – rule (4.32)		
it	n_c	*time*	*it*	n_c	*time*	*it*	n_c	*time*	*it*	n_c	*time*
$m_e = 1000$ $m_i = 1000$ $nac = 500$ rank$(G) = 4900$											
71 (391)	18	6.9	103 (756)	113	14.3	82 (403)	21	6.6	390 (1261)	195	24.8
$m_e = 2000$ $m_i = 2000$ $nac = 1000$ rank$(G) = 4900$											
90 (1184)	17	19.9	85 (1555)	87	26.0	71 (850)	14	16.5	156 (938)	71	21.3
$m_e = 3500$ $m_i = 1000$ $nac = 500$ rank$(G) = 4900$											
80 (528)	12	18.9	70 (665)	59	22.5	86 (560)	13	19.3	342 (1144)	169	40.8

Table 9

$n = 5000$ spars$(G) = 99.8\%$ $\|G\| = 1$ $K(G) = 100$ spars$((A^T\ C^T)^T) = 99.9\%$ $\|(A^T\ C^T)^T\| = 1$ $K((A^T\ C^T)^T) = 10$									
	VPM			AVPM			AFPM-10		
rank(Z^TGZ)	*it*	n_c	*time*	*it*	n_c	*time*	*it*	n_c	*time*
$m_e = 2000$ $m_i = 2000$ $nac = 1000$ rank$(G) = 4500$									
2000	80 (494)	11	13.8	97 (601)	22	16.5	102 (1145)	71	21.7
1750	83 (521)	12	14.2	74 (523)	18	14.7	94 (856)	62	20.3
1500	89 (399)	16	14.4	79 (421)	13	14.7	109 (937)	69	23.7
$m_e = 2000$ $m_i = 2000$ $nac = 1000$ rank$(G) = 4000$									
2000	86 (366)	18	11.8	81 (442)	30	13.6	92 (727)	65	18.4
1500	84 (831)	9	16.7	74 (857)	13	17.2	240(3118)	164	51.8
1000	98 (1023)	8	19.7	90 (995)	15	20.2	137 (2170)	93	36.5
$m_e = 2000$ $m_i = 2000$ $nac = 1000$ rank$(G) = 3500$									
2000	79 (327)	15	15.9	82 (409)	25	18.5	99 (727)	64	25.0
1250	87 (358)	20	13.7	67 (376)	25	14.2	92 (727)	65	21.6
500	76 (544)	9	13.8	81 (665)	15	15.7	91 (793)	60	19.5

constraints active in the solution. Then, Z^TGZ is the reduced Hessian matrix; when this matrix has a full column rank, problem (1.1) has a unique solution, otherwise, K^* is not a singleton. We may observe that, on these test problems, the effectiveness of the three methods appears independent on the rank of G.
Table 10 shows the results obtained by the VPM and the AVPM with rule (4.31) on some large–scale and very large–scale test problems of the CUTE library [6]. These results confirm that when, in the VPM and the AVPM, a convenient updating rule for the choice of the projection parameter is used, the corrective step along a descent direction or a projection arc arises in few iterations and the two schemes show an efficient numerical behaviour.

6.4 Numerical behaviour of parallel inner solvers in the projection–type methods

The numerical results reported in the following tables enables us to evaluate on a multiprocessor system the effectiveness of the class of the projection–type methods combined as a parallel iterative solver for the the solution of the subproblems (5.2).
In particular, we consider QP problems arising in constrained bivariate interpolation, (see Chapter 7) and we use a parallel implementation in Fortran 90 of the accelerated

Table 10 - Test problem CVXQP from CUTE

	AVPM – rule (4.31)			VPM – rule (4.31)		
n	it	n_c	$time$	it	n_c	$time$
$m_e = n/4$						
3000	310 (817)	1	5.6	308 (775)	3	5.0
5000	257 (460)	2	5.5	249 (425)	1	5.2
7000	247 (392)	1	7.8	264 (433)	1	8.4
9000	315 (882)	3	20.3	304 (784)	2	18.8
$m_e = n/2$						
3000	114 (204)	2	17.5	116 (218)	6	19.4
7000	101 (200)	3	54.2	103 (190)	3	50.3
9000	133 (420)	2	148.6	135 (350)	1	127.1
11000	135 (245)	0	107.2	135 (245)	0	107.0
13000	148 (232)	3	117.8	142 (229)	2	112.3
17000	129 (206)	6	159.6	128 (218)	3	172.2

splitting method (ASM) described in Chapter 3 (see (3.17)), with $D = diag(G_{ii})$, G_{ii} being the diagonal blocks of the matrix G.
Interprocessor communications have been performed by the Cray Shared Memory Access Library routines [8].
When we implement the ASM on a distributed memory system, the data of the problem must be distributed among the available processing elements (PEs) P_l, $l = 1,\dots,p$. A natural way to distribute the matrix G and the n–vector q is to allocate the nonzero entries of the j–th row of G and the j–th element of q in the local memory of P_l for $j = (l-1)s+1,\dots,l\cdot s,\ \ l = 1,\dots,p$, where $s = n/p$ (we assume p divisor of n).
If we use PSOR or PGPSOR as inner solver, we allocate in the local memory of P_l, $l = 1,\dots,p$, the nonzero entries of the j–th row of the matrix $\begin{pmatrix} C \\ A \end{pmatrix}$ and the j–th element of the ν–vector $\begin{pmatrix} d \\ b \end{pmatrix}$ for $j = (l-1)\nu/p+1,\dots,l\nu/p$, where $\nu = m_e + m_i$ (we also assume p divisor of ν).
The matrix M of (5.7) is partitioned as indicated in (5.8) with $\nu_l = \nu/p$, $l = 1,\dots,p$; the nonzero entries of $\bar{M}_l$ and g_l are allocated in the local memory of P_l.
If we use OPSOR as inner solver, we distribute among the PEs the nonzero entries of the submatrices $\bar{M}_l$, of the corresponding rows of $\begin{pmatrix} C \\ A \end{pmatrix}$ and vectors g and $\begin{pmatrix} d \\ b \end{pmatrix}$ as in PSOR and PGPSOR, but in this case $\nu_{2l-2} = \nu_{over}$, $l = 2,3,\dots,p$, and $\nu_{2l-1} = (\nu-(p-1)\nu_{over})/p$, $l = 1,2,\dots,p$. The parameter ν_{over} is chosen experimentally, so as to obtain the mimimum number of OPSOR iterations. In [19], numerical experiments show that a *good* value for ν_{over} is about $\nu_{over} = \nu/(2p-1)$.
Furthermore, local copies of y^k and of the current vector iterate of the parallel inner solver are allocated in the memory of all PEs. As for the parallel inner solver, at each ASM iteration, the j–th element of the current outer vector iterate is computed by P_l for $j = (l-1)s+1,\dots,l\cdot s,\ \ l = 1,\dots,p$; the remaining elements are updated by a synchronous multinode broadcast operation among the p PEs.
This static allocation of data among the p available PEs gives a well balanced workload. In fact in all the experiments concerning the three versions of ASM (ASM–PSOR, ASM–PGPSOR, ASM–OPSOR) we observe that idle time is negligible with respect to the total

Table 11

	ASM-PSOR		ASM-PGPSOR		ASM-OPSOR	
PEs	*it* (k_{it})	*time*	*it* (k_{it})	*time*	*it* (k_{it})	*time*
$n = 4000$, sparsity of $G = 99.65\%$ sparsity of $(C^T \ \ A^T) = 99.75\%$						
$m_i = 1750$, $m_e = 750$, sparsity of $M = 95.09\%$						
ASM-SOR	53(131)	31.47				
2	106(404)	28.72	62(134)	24.62	81(256)	28.59
4	113(611)	17.94	76(206)	16.28	101(308)	13.36
8	101(735)	9.38	71(264)	8.92	86(379)	6.89
16	100(776)	6.15	100(313)	6.70	97(411)	4.68
32	103(795)	3.99	75(320)	4.46	85(489)	3.47
64	107(816)	3.15	74(327)	3.63	99(469)	2.78
$m_i = 2500$, $m_e = 1000$, sparsity of $M = 95.10\%$						
ASM-SOR	62 (131)	59.94				
2	144 (900)	92.95	72 (237)	60.00	112(558)	82.79
4	144 (1400)	58.14	128 (445)	47.73	120(678)	32.82
8	133 (1656)	28.41	99 (490)	22.49	130(849)	17.17
16	132 (1704)	16.94	129 (563)	15.80	132(873)	10.48
32	130 (1803)	9.63	129 (589)	10.10	134(915)	7.06
64	136 (1841)	6.92	151 (642)	8.61	122(901)	5.23
$m_i = 4000$, $m_e = 2000$, sparsity of $M = 95.11\%$						
ASM-SOR	123(527)	143.34				
2	407(2950)	333.96	208(599)	151.60	200(1078)	198.17
4	234(4459)	200.74	358(1284)	144.68	407(2160)	96.25
8	252(5169)	94.82	211(1489)	68.56	235(3162)	55.86
16	236(5641)	57.19	259(2317)	61.03	203(3657)	34.83
32	224(5829)	28.73	213(2323)	34.35	237(3525)	22.42
64	232(5949)	18.37	232(2702)	30.84	211(3762)	14.08

elapsed time. For example, if we exclude the first step of the inner OPSOR iteration where one PE does not work, the maximum idle time for the three version of ASM is about 0.8% of the total elapsed time for $p = 8$, 1.3% for $p = 16$, 2.4% for $p = 32$.

The results reported in Table 10 shows the behaviour of ASM combined with PSOR, PGPSOR and OPSOR as inner solvers for a different number of PEs. The sparsity of the matrices G, C and A is very large. Consequently the matrix M is sparse (sparsity greater than 95%).

In Table 11 k_{it} denotes the total number of iterations of the inner solver. The serial version of ASM (executing on one PE) uses the projected SOR method as inner solver with a value of ω that provides the best result. This last inner solver has in general a better performance than the serial version of GPSOR: when the value of ω is a good estimate of its optimal value, GPSOR performs extra–iterations with respect to the SOR method. For example, for the problem in Table 11 with $n = 4000$, $\nu = 3500$ and $\omega = 1.2$, $k_{it} = 131$ for SOR and $k_{it} = 136$ for GPSOR. For an arbitrary value of ω, for instance $\omega = 0.9$, the number of GPSOR iterations is less than that of SOR method ($k_{it} = 190$ for SOR, $k_{it} = 157$ for GPSOR), but the complexity of GPSOR reduces its performance (*time* = 69.89 for SOR, *time* = 88.80 for GPSOR).

In the PSOR and OPSOR inner solvers the diagonal entries of the matrices E_i are computed as the upper bounds of the inequalities (5.10) and (5.12) respectively (reduced by a small constant). For the PGPSOR inner solver, we use $E_i = \omega I$ where ω approximates the value that provides the lowest total number of inner iterations.

Table 11 shows that, as p increases, the number of iterations of the three solvers increases, because the iterative schemes change with the number of blocks in which M is partitioned. Thus, we are not considering the same algorithm when p increases and, consequently, it is not very significant to discuss about the scalability or the speedup of the three

Table 12

	ASM-PSOR		ASM-PGPSOR		ASM-OPSOR	
PEs	*it* (k_{it})	*time*	*it* (k_{it})	*time*	*it* (k_{it})	*time*
$n = 4000$, $m_i = 2500$, $m_e = 1000$, sparsity of $G = 99.65\%$						
sparsity of $(C^T \quad A^T) = 99.50\%$ sparsity of $M = 81.78\%$						
ASM-SOR	66(205)	154.36				
2	147(1680)	444.32	84 (231)	160.27	139(1083)	371.96
4	145(2515)	294.89	112(426)	127.94	127(1323)	133.30
8	130(2977)	138.85	123(548)	65.44	146(1516)	65.23
16	133(3183)	74.08	102(591)	38.55	158(1590)	36.81
32	144(3355)	37.25	120(761)	23.22	157(1593)	20.87
64	130(3359)	21.27	95(814)	15.91	157(1645)	12.90
sparsity of $(C^T \quad A^T) = 99.38\%$ sparsity of $M = 73.02\%$						
ASM-SOR	66(224)	222.98				
2	122(2204)	768.79	81(237)	221.31	125(1384)	631.84
4	111(3265)	535.04	90(409)	168.11	140(1579)	220.24
8	115(3865)	254.66	90(495)	81.83	131(1913)	112.40
16	116(4156)	147.57	103(670)	58.34	119(2002)	61.89
32	110(4282)	64.29	103(790)	30.82	117(2116)	35.40
64	105(4359)	35.91	111(906)	21.59	118(2169)	20.78
sparsity of $(C^T \quad A^T) = 99.25\%$ sparsity of $M = 63.54\%$						
ASM-SOR	64(185)	241.26				
2	148(2663)	1172.93	81(250)	293.51	132(1632)	942.97
4	134(3945)	837.71	96(456)	236.61	130(2042)	363.15
8	126(4640)	404.81	92(656)	134.74	140(2308)	174.93
16	127(5077)	236.82	97(712)	79.01	139(2515)	98.92
32	126(5218)	99.71	91(740)	36.04	133(2552)	53.26
64	124(5251)	54.08	104(889)	25.49	124(2686)	31.19

schemes. We can only observe that, as expected, for a fixed p, the time to execute one iteration of OPSOR method is not much greater than that of the PSOR iteration, while one PGPSOR iteration is more than twice the PSOR iteration.
From Table 11 we point out that it is convenient to use a parallel inner solver when p is sufficiently large, generally $p > 2$. Nevertheless, only for large–scale problems it appears useful to have a great number of PEs (in the case $\nu = 2500$, see the elapsed times for 32 and 64 PEs).
If we compare the three methods we observe that:

- for $p > 2$, the ASM–OPSOR has better performance with respect to the others;
- as p increases, the ASM–PSOR becomes more convenient than the ASM–PGPSOR; this is justified by the high iteration cost of PGPSOR method with respect to PSOR method.

In the next experiment, we are interested in comparing the effectiveness of the three methods for different level of sparsity of the matrix M.
Table 12 shows the increasing performance of ASM–PGPSOR with respect the ASM–OPSOR as the sparsity of M decreases. This arises because, for a fixed p, the number of ASM–PGPSOR iterations does not strongly depend on the sparsity of M. On the other hand, when the number of nonzero entries of M increases, the upper bounds of the relaxation factors in the OPSOR method become so small to produce a slow convergence. We point out that the comparison between the two inner solvers depends also on the number of available PEs. In practice, the behaviour of ASM–OPSOR respect to ASM–PGPSOR is analogous to that of ASM–PSOR in the Table 11.
Table 13 allows to point out that the previous considerations depend on the choice of the diagonal matrices E_i in the PSOR and OPSOR inner solvers, especially when the

Table 13

	ASM-PSOR			ASM-PGPSOR			ASM-OPSOR		
PEs	ω	*it* (k_{it})	*time*	ω	*it* (k_{it})	*time*	ω	*it* (k_{it})	*time*
$n = 4000$, $m_i = 2500$, $m_e = 1000$, sparsity of $G = 99.65\%$									
sparsity of $(C^T \quad A^T) = 99.75\%$ sparsity of $M = 95.10\%$									
	$\bar\omega \approx 0.09$						$\bar\omega \approx 0.19$		
	0.25	132 (1704)	16.99	0.9	128 (692)	17.93	0.54	149 (580)	9.16
	0.39	144 (985)	12.28	1.1	118 (660)	17.29	0.89	166 (361)	8.03
16	0.45	146 (799)	11.03	1.3	141 (656)	17.50	0.90	215 (563)	9.44
	0.47	146 (754)	10.76	1.5	129 (563)	15.80	0.91	278 (2472)	20.60
	0.48	*	*	1.7	127 (599)	16.34	0.93	*	*
	$\bar\omega \approx 0.08$						$\bar\omega \approx 0.17$		
	0.22	129 (1616)	9.01	0.9	124 (680)	10.92	0.52	134 (600)	6.17
	0.33	143 (1089)	7.43	1.3	124 (644)	10.56	0.70	148 (410)	5.59
32	0.40	142 (936)	6.96	1.5	129 (589)	10.10	0.80	164 (321)	5.35
	0.45	145 (814)	6.60	1.7	115 (585)	9.91	0.85	169 (294)	5.29
	0.46	*	*	1.9	150 (623)	10.53	0.87	155 (5263)	22.09
sparsity of $(C^T \quad A^T) = 99.38\%$ sparsity of $M = 73.02\%$									
	$\bar\omega \approx 0.07$						$\bar\omega \approx 0.13$		
	0.20	128 (2099)	33.16	1.3	87 (815)	30.79	0.20	125 (2051)	33.94
	0.30	131 (1400)	23.80	1.4	100 (788)	30.20	0.30	126 (1265)	22.76
32	0.40	154 (1016)	18.79	1.5	103 (790)	30.82	0.50	152 (704)	15.21
	0.43	157 (930)	17.63	1.6	100 (780)	29.97	0.80	153 (362)	10.46
	0.44	*	*	1.7	118 (783)	30.40	0.85	*	*
	$\bar\omega \approx 0.06$						$\bar\omega \approx 0.12$		
	0.20	130 (2152)	19.16	0.7	92 (939)	21.55	0.30	126 (1313)	13.94
	0.30	130 (1374)	13.80	1.3	104 (872)	20.46	0.40	141 (927)	11.13
64	0.40	155 (1003)	11.17	1.5	111 (906)	21.59	0.60	147 (558)	8.46
	0.43	164 (1594)	15.33	1.7	126 (935)	21.97	0.80	146 (360)	6.92
	0.44	*	*	1.9	105 (881)	20.74	0.85	*	*

The character $*$ indicates that the inner solver does not converge.

sparsity of M decreases. The inequalities (5.10) and (5.12) provide sufficient conditions for the PSOR and OPSOR convergence respectively. But for this methods we can obtain a faster convergence also for diagonal matrices $E_i = \omega I$ with values of ω greater than the minimum of the upper bounds (5.10) and (5.12) ($\bar\omega$ in Table 13). This observation is in agreement with the considerations related to Table 1 about the projection parameter. On the contrary the total number of inner iterations of ASM–PGPSOR is weakly dependent on the value of ω.

In conclusion, for large and very sparse quadratic programs, the ASM–OPSOR appears the more efficient approach; for intermediate level of sparsity of M, when ω satisfies the PSOR and OPSOR convergence conditions, ASM–PGPSOR achieves a better performance.

CHAPTER 7

Two applications

In this chapter, we describe two applications, arising in the framework of data analysis, that can be efficiently solved by projection–type methods.

7.1 A Constrained Bivariate Interpolation Problem

Given N pairwise distinct and arbitrarily spaced points $P_i = (x_i, y_i)$, $i = 1, \dots, N$, in a domain $\mathcal{D}$ of the $x - y$ plane and N real numbers f_i, $i = 1, \dots, N$ one has to determine a function $f(x, y)$ of class C^1 in $\mathcal{D}$ whose values in P_i are exactly f_i, $i = 1, \dots, N$, and whose first or second order partial derivatives satisfy appropriate equality or inequality constraints on a given set of points in $\mathcal{D}$.
This problem can easily have thousands of interpolation points P_i.
A global approach for solving this problem consists of the following three steps.

1. *Triangulation.* The points P_i, are used as the vertices of a triangulation of the domain $\mathcal{D}$ (see [23], [1]).

2. *Curve Network.* The function $f(x, y)$ and its first order partial derivatives are defined on the subset consisting of the union of all edges of the triangulation. The curve network is obtained by interpolants which are constructed using the data f_i and the first order partial derivatives $f_x(P_i)$, $f_y(P_i)$ in the points P_i.

3. *Blending.* The function $f(x, y)$ is extended to $\mathcal{D}$ by means of a blending method which will assume arbitrary position and slope on the boundary of a triangle (see [2], [3], [22]).

In step 2., we can express the values of $f(x, y)$ and its first order partial derivatives in a point Q in the triangle of vertices P_i, P_j and P_k as linear combination of f_i, f_j, f_k, $f_x(P_i), f_y(P_i)$, $f_x(P_j), f_y(P_j), f_x(P_k)$ and $f_y(P_k)$. In order to obtain the $2N$-vector $x = (f_x(P_1), f_y(P_1), f_x(P_2), f_y(P_2), \dots, f_x(P_n), f_y(P_n))^T$, we consider the problem of computing a piecewise cubic interpolating curve which minimizes the functional

$$\sum_{ij \in \mathcal{N}_e} \int_{e_{ij}} \left(\frac{\partial^2 f}{\partial {e_{ij}}^2} \right)^2 ds_{ij},$$

where ds_{ij} is the element of the arc length along the edge e_{ij} in the triangulation with endpoints P_i and P_j, and $\mathcal{N}_e$ is a list of indices representing the edges of the triangulation.

In [35] Nielson proved that the vector x minimizes the quadratic function

$$\frac{1}{2}x^TGx + x^Tq, \tag{7.1}$$

where $G = (G_{ij})$, $q = (q_i)$, $i, j = 1, \ldots, N$ and

$$G_{ii} = \begin{pmatrix} \alpha_i & \beta_i \\ \beta_i & \gamma_i \end{pmatrix}, \quad G_{ij} = \begin{pmatrix} \alpha_{ij} & \beta_{ij} \\ \beta_{ij} & \gamma_{ij} \end{pmatrix} \quad \text{if} \quad ij \in \mathcal{N}_i;$$
$$\text{otherwise} \quad G_{ij} = 0,$$

with
$\alpha_{ij} = \frac{(x_j-x_i)^2}{2\|e_{ij}\|^3}$, $\beta_{ij} = \frac{(x_j-x_i)(y_j-y_i)}{2\|e_{ij}\|^3}$, $\gamma_{ij} = \frac{(y_j-y_i)^2}{2\|e_{ij}\|^3}$, $\alpha_i = 2\sum\alpha_{ij}$, $\beta_i = 2\sum\beta_{ij}$, $\gamma_i = 2\sum\gamma_{ij}$; and

$$q_i = -\frac{3}{2}\left(\sum\frac{(f_j-f_i)(x_j-x_i)}{\|e_{ij}\|^3}, \quad \sum\frac{(f_j-f_i)(y_j-y_i)}{\|e_{ij}\|^3}\right)^T.$$

Here, $\mathcal{N}_i = \{ij : e_{ij}$ is the edge of the triangulation with endpoint $P_i\}$, $\|e_{ij}\|$ is the length of e_{ij} and $\sum$ is the summation over indices $ij \in \mathcal{N}_i$.

Theorem **4** *The symmetric matrix G of the problem (7.1) is positive definite.*

Proof. See [16].
The constraints on the surface $f(x, y)$ can be expressed as a set of linear equalities and inequalities of the form

$$\begin{aligned} Cx &= d \\ Ax &\geq b. \end{aligned} \tag{7.2}$$

In conclusion, to determine the curve network we must solve the strictly convex quadratic programming (7.1)-(7.2) where G is positive definite and, in general, quite sparse.
If we set $D = diag(G_{11}, \ldots, G_{NN})$ and $H = G - D$, we can prove that $D + H$ is a P–regular splitting of G and we can obtain the solution of (7.1)-(7.2) by using the SM.

Theorem **5** *The symmetric matrix $D - H$ is positive definite.*

Proof. If $w = (w_1^T, w_2^T, \ldots, w_N^T)^T$ with $w_i = (\Psi_x^i, \Psi_y^i)^T$ is any vector different from zero, we must prove that $w^T(D - H)w > 0$, i.e.

$$\begin{aligned} w^T(D-H)w &= \sum_{i=1}^N w_i^T(G_{ii}w_i - \sum_{ij\in\mathcal{N}_i} G_{ij}w_j) \\ &= \sum_{i=1}^N \sum_{ij\in\mathcal{N}_i} \frac{1}{\|e_{ij}\|^3}[((x_j-x_i)\Psi_x^i + (y_j-y_i)\Psi_y^i)^2 \\ &\quad - \frac{1}{2}((x_j-x_i)\Psi_x^i + (y_j-y_i)\Psi_y^i)((x_j-x_i)\Psi_x^j + (y_j-y_i)\Psi_y^j)]. \end{aligned}$$

Now, for each cubic Hermite polynomial φ_{ij} defined on e_{ij} such that in the points P_i and P_j $\varphi_i = \varphi_j = 0$ and $\varphi_i' = (x_j-x_i)\Psi_x^i + (y_j-y_i)\Psi_y^i$ and $\varphi_j' = -(x_j-x_i)\Psi_x^j - (y_j-y_i)\Psi_y^j$, that is, $\varphi_{ij}(t) = t(1-t)^2\varphi_i' + t^2(t-1)\varphi_j'$, $(0 \leq t \leq 1)$, we have

$$\begin{aligned} \sum_{ij\in\mathcal{N}_e} \frac{1}{\|e_{ij}\|^3}\int_0^1 (\varphi_{ij}''(t))^2 dt &= \sum_{ij\in\mathcal{N}_e} \frac{1}{\|e_{ij}\|^3}(\varphi_{ij}''(1)\varphi_{ij}'(1) - \varphi_{ij}''(0)\varphi_{ij}'(0)) \\ &= -\sum_{i=1}^N \sum_{ij\in\mathcal{N}_i} \frac{1}{\|e_{ij}\|^3}\varphi_{ij}''(0)\varphi_{ij}'(0) \\ &= 4w^T(D-H)w. \end{aligned}$$

We have used the relations $\varphi'_{ij}(1) = \varphi'_{ji}(0)$ and $\varphi''_{ij}(1) = -\varphi''_{ji}(0)$. Since there always exists some e_{ij} on which $\varphi''_{ij}(t) \neq 0$, we have proved that $w^T(D - H)w > 0$.

An extensive experimentation on test problems arising from this application shows the effectiveness of the SM combined by the projected SOR method as inner solver for medium and large–scale problems [17]. In Table 14, the results obtained by solving test problems of this kind by the SM with $D = diag(G_{ii})$ are compared with those obtained by the MPM and the VPM with $D = diag(G_{ii})$ (as in the SM) and $D = I$. We observe that in this application the effectiveness of the SM is essentially the same of that of the VPM with $D = diag(G_{ii})$.

Table 14. Special quadratic programs arising in constrained bivariate interpolation problems ($tol = 10^{-6}$).

		SM (SOR)		VPM (SOR)				MPM (SOR)	
		$D = diag\{G_{ii}\}$		$D = diag\{G_{ii}\}$		$D = I$			
m_e	m_i	*it*	*time*	*it*	*time*	*it*	*time*	*it*	*time*
$n = 1000$		spars(G)=98.6%		$K(G) = 105.3$		$\|G\| = 839.3$		spars($(C^T\ A^T)^T$)=99.1%	
50	50	53 (110)	0.09	42 (94)	0.09	156 (685)	0.3	277 (1650)	0.4
200	300	40 (133)	0.8	42 (180)	1.0	147 (961)	3.8	193 (1928)	7.1
400	600	34 (232)	5.3	44 (352)	7.9	85 (1036)	15.3	145 (3022)	44.2
$n = 3000$		spars(G)=99.6%		$K(G) = 146.5$		$\|G\| = 2874.6$		spars($(C^T\ A^T)^T$)=99.7%	
200	100	216 (435)	1.3	116 (190)	1.0	207 (1226)	1.7	301 (2520)	2.0
500	500	234 (434)	5.5	116 (228)	3.3	197 (1232)	6.9	265 (2520)	11.3
1000	1000	106 (312)	12.6	95 (261)	10.8	156 (1231)	28.5	238 (3400)	72.2

These results are carried out on a Digital Alpha 500/333 Mhz.

7.2 Training Support Vector Machines

We consider the numerical solution of the QP problem arising in training learning machines, named Support Vector Machines (SVMs) [7], [51].
The learning technique SVM performs pattern recognition between two point classes by finding a decision surface determined by certain points of the training set.
We briefly sketch the SVM technique starting from the simple case of two linearly separable classes (see [40], [42]).
We assume that we have a data set (training set) of labeled examples

$$D = \{(p_i, y_i),\ i = 1, \dots, n, \quad p_i \in R^m,\ y_i \in \{-1, 1\}\}$$

and we wish to determine the hyperplane that separates the data and leaves the maximum margin between the two classes, where the margin is defined as the sum of the distances of the hyperplane from the closest points of the two classes. When the two classes are nonseparable we can determine the hyperplane that maximizes the margin and minimizes a quantity proportional to the number of misclassification errors. The trade-off between the largest margin and the lowest number of errors is controlled by a positive constant C that has to be chosen beforehand. The solution to this problem is a linear classifier [42]

$$F(p) = sign\left(\sum_{i=1}^{n} x_i y_i p^T p_i + b\right)$$

whose coefficients x_i are the solution of the following quadratic programming problem

$$\begin{aligned} &\textit{minimize} \quad \tfrac{1}{2}x^T Gx - \sum_{i=1}^{n} x_i \\ &\textit{subject to} \quad \sum_{i=1}^{n} y_i x_i = 0, \qquad 0 \le x_j \le C, \qquad j = 1, \ldots, n, \end{aligned} \tag{7.3}$$

where $x = (x_1, x_2, \ldots, x_n)^T$ and the entries G_{ij} of G are defined by $G_{ij} = y_i y_j p_i^T p_j$. Thus, the classifier is determined by the data points associated to the nonzero coefficients. These data points, termed support vectors (SV), condense the information of the training set sufficient to classify new data points.
The previous technique can be extended to the general case of nonlinear separating surfaces. This is easily done by mapping the input points into a space Z, called feature space, and by formulating the linear classification problem in the feature space. Typically Z is a Hilbert space of finite or infinite dimension. If $p \in R^m$ is an input point, let $\varphi(p)$ be the corresponding feature point with φ a mapping from R^m to Z. The solution to the classification problem in the feature space will have the following form

$$F(p) = sign\left(\sum_{i=1}^{n} x_i y_i \varphi(p)^T \varphi(p_i) + b\right)$$

and therefore will be nonlinear in the original input variables. In this case, the coefficients x_i are the solution of a QP problem of the form (7.3) where $G_{ij} = y_i y_j \varphi(p_i)^T \varphi(p_j)$. At first sight it might seem that the nonlinear separating surface cannot be determined unless the mapping φ is completely known. However, since φ enters only in scalar product between feature points, if we find an expression for the scalar product in feature space which uses the points in input space only, that is

$$\varphi(p_i)^T \varphi(p_j) = K(p_i, p_j), \tag{7.4}$$

full knowledge of φ is not necessary. The symmetric function K in (7.4) is called kernel. We may conclude that the extension of the theory to the nonlinear case is reduced to finding kernels which identify certain families of decision surfaces and satisfy equation (7.4). When such kernel is identified, the nonlinear separating surface can be found by computing the solution $(x_1, \ldots, x_n)^T$ of the QP problem (7.3) with

$$G_{ij} = y_i y_j K(p_i, p_j) \tag{7.5}$$

and the classification stage can be performed by evaluating

$$sign\left(\sum_{i=1}^{n} x_i y_i K(p_i, p) + b\right).$$

Two frequently used kernels are the polynomial kernel, $K(p_i, p_j) = (1 + p_i^T p_j)^d$, and the Gaussian kernel, $K(p_i, p_j) = exp(-\|p_i - p_j\|^2/(2\sigma^2)$. The separating surface in input space is a polynomial surface of degree d for the polynomial kernel and a weighted sum of Gaussians centered on the support vectors for the Gaussian kernel.
From the computational point of view, the main effort in the implementation of a SVM consists in solving the linearly constrained convex QP problem (7.3). The size of this problems is equal to the number of points in the training set and, consequently, in many interesting applications of the SVMs we must to solve a large–scale QP problem

(n larger than a few thousands [40], [43]). In these cases, since the Hessian matrix G is generally with a large number of nonzero elements, it is very difficult to solve (7.3) without some kind of decomposition technique. A special decomposition algorithm for solving this problem is proposed in [39]. Exploiting the particular form of the objective function, this method finds the solution of the full problem (7.3) by solving iteratively QP subproblems of (7.3) which depend only on $n1$ variables ($n1 \ll n$). Each subproblem consists in minimizing the same objective function of (7.3) with respect to the selected $n1$ components, the others being kept constant. The structure of the constraints of each subproblem is as in the original problem: box constraints and only one equality constraint.
Thus, the effectiveness of this decomposition scheme for solving very large–scale problems, is strictly dependent on the effectiveness of the solver used for the inner QP subproblems of smaller size.
In [52] the VPM described in Section 4.1 is used as solver for the QP problems arising in training SVMs with Gaussian kernel. This method requires explicit storage of G and then applies to all cases in which the size of the problem allows to keep this matrix in memory. When the size of the problem is so large to force the use of the above decomposition technique, the VPM can be useful for solving the smaller inner QP subproblems.
The application of a projection–type method in this particular context is suggested by the following two reasons. First of all, the special nature of the Hessian matrix of this problem,

$$G_{i,j} = y_i y_j exp\left(\frac{-\|p_i - p_j\|^2}{2\sigma^2}\right) \qquad \sigma \in R,$$

suggests that, for sufficiently small values of σ, a good convergence rate can be expected (the matrix G *tends* to the identity matrix). Second, the special structure of the constraints implies that, if in a projection–type scheme the Hessain matrix D of the inner QP problems is a diagonal matrix, each subproblem becomes a single constrained separable quadratic program subject to upper and lower bounds. For this kind of problems very efficient algorithms are available. Furthermore, by using the VPM it is possible to have a good convergence rate also for the nonextremely small values of σ that must be used for training support vector machines with high classification accuracy. This last feature is obtained by using the following updating rule for the variable projection parameter ρ_k:

$$\rho_k = \begin{cases} \rho_{k-1} & \text{for } \|Gd^{k-1}\|^2 \le \epsilon \|d^{k-1}\|^2, \\ \begin{cases} \frac{d^{k-1^T} D d^{k-1}}{d^{k-1^T} G d^{k-1}} & \text{if } \mathrm{mod}(k,6) \ge 3, \\ \frac{d^{k-1^T} G d^{k-1}}{d^{k-1^T} G D^{-1} G d^{k-1}} & \text{otherwise}, \end{cases} & \text{otherwise.} \end{cases} \tag{7.6}$$

The numerical experiments reported in [52] show that, on test problems arising in training SVMs with Gaussian kernel, this particular updating rule is more efficient than the rule (4.31) or (4.32).
In the following we report the results of a set of numerical experiments that show the computational behaviour of the VPM on test problems arising in training support vector machines with Gaussian kernel. We compare this scheme with another projection–type method, the projection method with limited minimization rule (denoted by FPM as in Section 6.3), and with the active set method implemented in the routine E04NCF of the NAG library. This last routine is designed to solve dense QP problems.
The experiments are carried out on a Digital Personal Workstation 500au, using the double precision (macheps= $2.22 \cdot 10^{-16}$) and a set of programs written in Fortran 77.

Two sets of QP test problems are considered. The first set (TP1) is obtained from an application of SVMs to a classical problem of computer vision: the recognition of 3-D objects from a single image. To this purpose, we have used images of objects from the COIL (Columbia Object Images Library) database [43]. The second set (TP2) consists in test problems arising in training a support vector machine for pattern recognition between two classes of points randomly generated in R^2 $(p = (p1, p2),\quad p1, p2 \in]0, 1[)$. In both cases we solve several test problems corresponding to different values of the parameter σ of the Hessian matrix G.
In the VPM and FPM we use the identity matrix as Hessian matrix of the subproblems $(D = I)$ and solve the corresponding separable subproblems by the algorithm proposed in [41]. Furthermore, since we know that the solution has many zero components, the matrix-vector product Gy^k required at each iteration is computed by working only with the nonzero components of y^k. The iterative procedures are terminated when the Karush–Kuhn–Tucker conditions are fulfilled within a tolerance of 10^{-6}.
In the following tables we denote by *it* and *time* the number of iterations and the time in seconds required by the methods; furthermore, we indicate by SV the number of support vectors (number of nonzero components in the solution of (7.3)).

Table 15

Test Problem TP1			$n = 1120$		$C = 4$	
		FPM			VPM	
σ	SV	ρ	*it*	*time*	*it*	*time*
350	921	2.2	846	19.1	284	6.1
750	538	0.7	1076	15.6	386	5.7
950	413	0.9	1396	17.1	451	5.3

The first experiments concern the comparison between the VPM and the FPM. For the FPM we report the results obtained with the empirical optimal value of the parameter ρ. Table 15 shows the better performance of the VPM.
In the second set of experiments we compare the numerical effectiveness of the VPM with the routine E04NCF of the NAG library.

Table 16

Test Problem TP1		$n = 1120$		$C = 4$
		E04NCF	VPM	
σ	SV	time	it	time
550	760	59.1	402	7.2
750	538	58.6	386	5.7
950	413	55.1	451	5.3

From Table 16 and Table 17 we may observe that the VPM is more efficient than the NAG routine and is slightly dependent on the value of σ.
In Table 18 different values for the size of the QP problems are considered. We observe that also for increasing size problems ($n = 2000$, $n = 3000$) the VPM appears very efficient in terms of number of iterations and computational time while the NAG routine becomes prohibitively expensive.
These results show that the VPM is an efficient approach for the medium to large size problems arising in training support vector machines with Gaussian kernel. Furthermore, when the particular application of the SVM requires to solve a very large–scale QP

Table 17

Test Problem TP2		$n = 1000$		$C = 2$
		E04NCF	VPM	
σ	SV	time	it	time
0.02	754	41.5	301	5.1
0.05	384	52.5	482	5.5
0.08	346	45.3	301	3.2
0.1	349	37.9	492	5.6

Table 18

Test Problem TP2		$\sigma = 0.08$		$C = 2$
		E04NCF	VPM	
n	SV	time	it	time
500	225	7.2	311	1.4
1000	346	45.3	301	3.2
2000	687	280.1	617	24.6
3000	981	707.0	648	58.2

problem and the decomposition scheme proposed in [39] must be used, the VPM can be an effective solver for the smaller inner QP subproblems.

Appendix

A large part of the programs by which the numerical experiments reported in Chapter 6 have been carried out, are available at the following URLs:
http://www.unife.it/AnNum97/index2.html
All the programs are written in Fortran 77 for a Digital Alpha workstation with the goal to evaluate the methods described in this work. An exhaustive documentation of each program is reported in the comments of the source file. In the following, we list the names of files and we give a brief description of their content:

- **test98.f**: the file contains the source of a program that generates a sparse linearly constrained convex QP problems with assigned features;
- **inputtest98**: the file contains an example of input data that must be supplied by the user for the program test98.f;
- **outputtest98**: the file shows the results obtained by the program test98.f when the user supplies as input data the values in the file inputtest98;
- **pm_sm.f**: the file contains the source of a program that computes a numerical solution of a sparse linearly constrained convex QP problems by the classical projection method or by the classical splitting method;
- **inputpm_sm**: the file contains an example of input data that must be supplied by the user for the program pm_sm.f;
- **pm_sm.dat**: the file shows the results obtained by the program pm_sm.f when the user supplies as input data the values in the file inputpm_sm;
- **mpm.f**: the file contains the source of a program that computes a numerical solution of a sparse linearly constrained convex QP problems by the modified projection method of Solodov and Tseng [49];
- **inputmpm**: the file contains an example of input data that must be supplied by the user for the program mpm.f;
- **mpm.dat**: the file shows the results obtained by the program mpm.f when the user supplies as input data the values in the file inputmpm;
- **fpm_asm.f**: the file contains the source of a program that computes a numerical solution of a sparse linearly constrained convex QP problems by the scaled gradient projection method with a limited minimization rule [4] or by the accelerated splitting method [18];
- **inputfpm_asm**: the file contains an example of input data that must be supplied by the user for the program fpm_asm.f;

- **fpm_asm.dat**: the file shows the results obtained by the program fpm_asm.f when the user supplies as input data the values in the file inputfpm_asm;
- **afpm.f**: the file contains the source of a program that computes a numerical solution of a sparse linearly constrained convex QP problems by the scaled gradient projection method with an Armijo rule along a projection arc [4];
- **inputafpm**: the file contains an example of input data that must be supplied by the user for the program afpm.f;
- **afpm.dat**: the file shows the results obtained by the program afpm.f when the user supplies as input data the values in the file inputafpm;
- **vpm.f**: the file contains the source of a program that computes a numerical solution of a sparse linearly constrained convex QP problems by the variable projection method;
- **inputvpm**: the file contains an example of input data that must be supplied by the user for the program vpm.f;
- **vpm.dat**: the file shows the results obtained by the program vpm.f when the user supplies as input data the values in the file inputvpm;
- **avpm.f**: the file contains the source of a program that computes a numerical solution of a sparse linearly constrained convex QP problems by the adaptive variable projection method;
- **inputavpm**: the file contains an example of input data that must be supplied by the user for the program avpm.f;
- **avpm.dat**: the file shows the results obtained by the program avpm.f when the user supplies as input data the values in the file inputavpm;
- **nag_test.f**: the file contains the source of a program that computes a numerical solution of a sparse linearly constrained convex QP problems by the NAG routine E04NKF;
- **inputnag**: the file contains an example of input data that must be supplied by the user for the program nag_test.f;
- **nag.dat**: the file shows the results obtained by the program nag_test.f when the user supplies as input data the values in the file inputnag.

Bibliography

[1] H. Akima, *A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points*, ACM Trans. Math. Software, **4**, (1978), pp. 148–159.

[2] R. E. Barnhill - G. Birkhoff - W. J. Gordon, *Smooth Interpolation in Triangles*, J. Approx. Theory, **8**, (1973), pp. 114–128.

[3] R. E. Barnhill - J. A. Gregory, *Compatible Smooth Interpolation in Triangles*, J. Approx. Theory, **15**, (1975), pp. 214–225.

[4] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, MA (1995).

[5] D. P. Bertsekas - J. N. Tsitsiklis, *Parallel and Distributed Computation, Numerical Methods*, Prentice–Hall, Englewood Cliffs, (1989).

[6] I. Bongartz - A. R. Conn - N. Gould - P. L. Toint, *CUTE: Constrained and Unconstrained Testing Environment*, ACM Trans. Math. Software, **21**, (1995), pp. 123–160.

[7] C. Cortes - V. N. Vapnik, *Support Vector Network*, Machine Learning, **20**, (1995), pp. 1–25.

[8] *CRAY T3E Fortran Optimization Guide, Cray Research Inc., SG–2518*, (1996).

[9] C. W. Cryer, *The Solution of a Quadratic Programming Problem Using Systematic Overrelaxation*, SIAM J. Control, **9**, (1971), pp. 385–392.

[10] S. Dafermos, *Traffic Equilibrium and Variational Inequalities*, Transportation Sci., **14**, (1980), pp. 42–54.

[11] A. R. De Pierro - A. N. Iusem, *Convergence Properties of Iterative Methods for Symmetric Positive Semidefinite Linear Complementarity Problems*, Math. Oper. Res., **18**, (1993), pp. 317–333.

[12] N. Dyn - J. Ferguson, *The Numerical Solution of Equality–Constrained Quadratic Programming Problems*, Mathematics of Computation, **41**, (1983), pp. 165–170.

[13] D. J. Evans, *The Use of Preconditioning in Iterative Methods for Solving Linear Equations with Symmetric Positive Definite Matrices*, J. IMA, **4**, (1968), pp. 295–314.

[14] C. S. Fisk - S. Nguyen, *Solution Algorithms for Network Equilibrium Models with Asymmetric User Costs*, Transportation Sci., **16**, (1980), pp. 361–381.

[15] E. GALLIGANI - V. RUGGIERO, *The Arithmetic Mean Preconditioner for Multivector Computers*, Intern. J. Computer Math., **44**, (1992), pp. 207–222.

[16] E. GALLIGANI, *C^1 surface interpolation with constraints*, Numer. Algorithms, **5**, (1993), pp. 549–555.

[17] E. GALLIGANI - V. RUGGIERO - L. ZANNI, *Splitting Methods for Quadratic Optimization in Data Analysis*, Intern. J. Comput. Math., **63**, (1997), pp. 289–307.

[18] E. GALLIGANI - V. RUGGIERO - L. ZANNI, *Splitting Methods for Constrained Quadratic Programs in Data Analysis*, Computers Math. Applic., **32**, (1996), pp. 1–9.

[19] E. GALLIGANI - V. RUGGIERO - L. ZANNI *Splitting Methods and Parallel Solution of Constrained Quadratic Programs*, Equilibrium Problems with Side Constraints. Lagrangean Theory and Duality II (F. Giannessi, A. Maugeri and M. De Luca eds.), Rend. Circ. Matem. Palermo, Ser. II Suppl. 48, (1997), pp. 121–136.

[20] E. GALLIGANI - V. RUGGIERO - L. ZANNI, *Parallel Solution of Large Scale Quadratic Pprograms*, High Performance Algoritms and Software in Nonlinear Optimization, (R. De Leone, A. Murli, P. Pardalos and G. Toraldo, Eds.), Applied Optimization 24, Kluwer Academic Publishers, Dordrecht, (1998), pp. 189–205.

[21] B. HE, *Solving a Class of Linear Projection Equations*, Numer. Math., **68**, (1994), pp. 71–80.

[22] I. M. KLUCEWICZ, *A Piecewise C^1 Interpolant to Arbitrarily Spaced Data*, Comput. Graph. Image Process., **8**, (1978), pp. 92–112.

[23] C. L. LAWSON, *Software for C^1 Surface Interpolation*, Mathematical Software III, J.R. Rice ed., Academic Press, (1977).

[24] W. LI, *Remarks on Convergence of the Matrix Splitting Algorithm for the Symmetric Linear Complementarity Problem*, SIAM J. Optim., **3**, (1993), pp. 155–163.

[25] Y. Y. LIN - J. S. PANG, *Iterative Methods for Large Convex Quadratic Programs: a Survey*, SIAM J. Control Optim., **25**, (1987), pp. 383–411.

[26] Z. Q. LUO - P. TSENG, *On the Convergence of a Matrix–Splitting Algorithm for the Symmetric Monotone Linear Complementarity Problem*, SIAM J. Control Optim., **29**, (1991), pp. 1037–1060.

[27] Z. Q. LUO - P. TSENG, *Error Bounds and Convergence Analysis of Matrix Splitting Algorithms for the Affine Variational Inequality Problem*, SIAM J. Optim., **2**, (1992), pp. 43–54.

[28] Z. Q. LUO - P. TSENG, *On the Linear Convergence of Descent Methods for Convex Essentially Smooth Minimization*, SIAM J. Control Optim., **30**, (1992), pp. 408–425.

[29] O. L. MANGASARIAN, *Solution of Symmetric Linear Complementarity Problems by Iterative Methods*, J. Optim. Theory Appl., **22**, (1977), pp. 465–485.

[30] O. L. MANGASARIAN, *Convergence of Iterates of an Inexact Matrix Splitting Algorithm for the Symmetric Monotone Linear Complementarity Problem*, SIAM J. Optim., **1**, (1991), pp. 114–122.

[31] O. L. MANGASARIAN - R. DE LEONE, *Parallel Successive Overrelaxation Methods for Symmetric Linear Complementarity Problems and Linear Programs*, J. Optim. Theory Appl. **54**, (1987), pp. 437–446.

[32] O. L. MANGASARIAN - R. DE LEONE, *Parallel Gradient Projection Successive Overrelaxation for Symmetric Linear Complementarity Problems and Linear Programs*, Ann. Oper. Res., **14**, (1988), pp. 41–59.

[33] P. MARCOTTE - J. H. WU, *On the Convergence of Projection Methods: Application to the Decomposition of Affine Variational Inequalities*, J. Optim. Theory Appl., **85**, (1995), pp. 347–362.

[34] *NAG Fortran Library Manual., Mark 18*, (1998).

[35] G. M. NIELSON, *A Method for Interpolating Scattered Data Based upon a Minimum Norm Network*, Math. Comp., **40**, (1983), pp. 253–271.

[36] S. N. NIELSEN - S. A. ZENIOS, *Massively Parallel Algorithms for Single Constrained Convex Programs*, ORSA J. on Computing, **4**, (1992), pp. 166–181.

[37] J. M. ORTEGA, *Numerical Analysis: A Second Course*, Academic Press, New York, (1972).

[38] J. M. ORTEGA - W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, (1970).

[39] E. OSUNA - R. FREUND - F. GIROSI, *An Improved Training Algorithm for Support Vector Machines*, Proceedings of the IEEE Workshop on Neural Networks and Signal Processing, (J. Principe, L. Giles, N. Morgan, E. Wilson eds.), Amelia Island, FL, (1997), pp. 276–285.

[40] E. OSUNA - R. FREUND - F. GIROSI, *Training Support Vector Machines: an Application to Face Detection*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico, (1997), pp. 130–136.

[41] P. M. PARDALOS - N. KOVOOR, *An Algorithm for a Single Constrained Class of Quadratic Programs Subject to Upper and Lower Bounds*, Math. Programming, **46**, (1990), pp. 321–328.

[42] M. PONTIL - A. VERRI, *Properties of Support Vector Machines*, Neural Computation, **10**, (1998), pp. 977–996.

[43] M. PONTIL - S. ROGAI - A. VERRI, *Support Vector Machines: A Large Scale QP Problem*, High Performance Algoritms and Software in Nonlinear Optimization, (R. De Leone, A. Murli, P. Pardalos and G. Toraldo, eds.), Applied Optimization 24, Kluwer Academic Publishers, Dordrecht, (1998), pp. 315–336.

[44] V. RUGGIERO - L. ZANNI, *On a Class of Iterative Methods for Large–Scale Convex Quadratic Programs*, Numerical Methods in Optimization, (A. Maugeri and E. Galligani, eds.), Rend. Circ. Matem. Palermo, Ser. II, Suppl. 58 , (1999), pp. 213–227.

[45] V. RUGGIERO - L. ZANNI, *On the Efficiency of Splitting and Projection Methods for Large Strictly Convex Quadratic Programs*, Nonlinear Optimization and Related Topics, (G. Di Pillo and F. Giannessi, eds.), Kluwer Academic Publ. B. V., (1999), pp. 401–413.

[46] V. Ruggiero - L. Zanni, *A Modified Projection Algorithm for Large Strictly Convex Quadratic Programs*, J. Optim. Theory Appl. , **104**, (2000), pp. 255–279.

[47] V. Ruggiero - L. Zanni, *An Overview on Projection–Type Methods for Convex Large–Scale Quadratic Programs*, Equilibrium Problems and Variational Models (F. Giannessi, A. Maugeri and P. Pardalos, eds.), Kluwer Academic Publishers, Dordrecht, (2000), to appear.

[48] V. Ruggiero - L. Zanni, *Variable Projection Methods for Large Convex Quadratic Programs*, Recent Trends in Numerical Analysis, (L.Brugnano and D.Trigiante, eds.), Advances in Computation: Theory and Practice, (2000), to appear.

[49] M. V. Solodov - P. Tseng, *Modified Projection-Type Methods for Monotone Variational Inequalities*, SIAM J. Control Optim., **34**, (1996), pp. 1814–1830.

[50] G. W. Stewart, *The Efficient Generation of Random Orthogonal Matrices with an Application to Condition Estimators*, SIAM J. Numer. Anal., **17**, (1980), pp. 403–409.

[51] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, (1995).

[52] A. Verri - L. Zanni, *Numerical Solution of Large Quadratic Programs in Training Support Vector Machines*, submitted.