VARIABLE PROJECTION METHODS FOR LARGE–SCALE QUADRATIC OPTIMIZATION IN DATA ANALYSIS APPLICATIONS

Emanuele Galligani
Department of Mathematics
University of Modena and Reggio Emilia
Via Campi 213/B, 41100 Modena, Italy
galligani@unimo.it

Valeria Ruggiero
Department of Mathematics
University of Ferrara
Via Machiavelli 35, 44100 Ferrara, Italy
rgv@dns.unife.it

Luca Zanni
Department of Mathematics
University of Modena and Reggio Emilia
Via Campi 213/B, 41100 Modena, Italy
zanniluca@unimo.it

Abstract: This paper concerns with the numerical evaluation of the variable projection method for quadratic programming problems in three data analysis applications. The three applications give rise to large–scale quadratic programs with remarkable differences in the Hessian definition and/or in the structure of the constraints.

Keywords: Large–scale quadratic programming, variable projection method, support vector machines, image restoration problem, bivariate interpolation problem.
1. Introduction

Consider the linearly constrained convex quadratic programming (QP) problem:

\[\begin{align*}
\text{minimize} & \quad f(x) = \frac{1}{2}x^T G x + q^T x \\
\text{subject to} & \quad C x = d, \quad A x \geq b,
\end{align*}\]  

(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 \(K = \{x \in \mathbb{R}^n \mid C x = d; A x \geq b\}\) is a nonempty polyhedral set and that \(f(x)\) is bounded from below on \(K\).

In some recent papers ([42], [43]), for the solution of large–scale QP problems of the form (1.1) we propose the Variable Projection Method (VPM), that includes as special cases the classical projection and splitting methods and the scaled gradient projection methods [6]. The VPM has the following iterative scheme (see also [44], [45] for theoretical and numerical features about some variants of the method):

1. let \(D\) be a symmetric positive definite matrix, \(x^{(0)}\) be an arbitrary vector and \(\rho_1\) be an arbitrary positive constant; \(k \leftarrow 1;\)
2. compute the unique solution \(y^{(k)}\) of the subproblem

\[\begin{align*}
\text{minimize} & \quad \frac{1}{2}x^T D \rho_k x + (q + (G - D \rho_k)x^{(k-1)})^T x \\
\text{subject to} & \quad C x = d, \quad A x \geq b;
\end{align*}\]  

(1.2)

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 \(\theta_k\) of the problem

\[\begin{align*}
\min \{ f(x^{(k-1)} + \theta d^{(k)}); \quad \theta \in (0, 1] \};
\end{align*}\]  

(1.3)

else

\[\theta_k = 1;\]
5. compute

\[x^{(k)} = x^{(k-1)} + \theta_k d^{(k)};\]  

(1.4)

6. terminate if \(x^{(k)}\) satisfies a stopping rule, otherwise update \(\rho_{k+1}\) by an appropriate rule; then \(k \leftarrow k + 1\) and go to step 2.
The R–linear convergence of the sequence \( \{x^{(k)}\} \) generated by the VPM to a solution of (1.1) is proved in [45] under the very general hypotheses that \( D \) is a symmetric positive definite matrix and the sequence \( \{\rho_k\} \) is bounded from below and above by positive constants.

The iterative scheme of the VPM requires essentially the matrix–vector product \( Gy^{(k)} \) and the solution of the QP subproblem (1.2) at each iteration. Of course, from the practical point of view, the matrix \( D \) must be an easily solvable matrix (diagonal or block diagonal matrix) in order to make each QP subproblem easier than the original problem. As consequence, the VPM turns the complexity of the original problem to the choice of a solver for separable (or nearly separable) QP problems which is specialized for the structure of the feasible region (see [32], [38] as examples of solvers for separable quadratic programs with special feasible region defined by box constraints and a single equality constraint). 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 [10], [20], [21], [28], [29]. Another important consideration is concerned with the choices of \( D \) and of the sequence of projection parameters \( \{\rho_k\} \).

Special choices of \( D \) and of the scalar parameters 1 give rise to the classical splitting and projection–type methods that are particular cases of the general VPM scheme.

The possibility to use a variable projection parameter at each step can be exploited to decrease the number of iterations and, consequently, to overcome the slow convergence rate exhibited by other splitting and projection–type methods (see [43]). To this aim, the following nonexpensive and efficient updating rules for \( \rho_k, k = 2, 3, \ldots \) are devised on the basis of heuristic considerations in [44]:

\[
\rho_k = \begin{cases} 
\rho_k-1 & \text{for } \|Gd^{(k-1)}\|^2 \leq \psi\|d^{(k-1)}\|^2 \\
\frac{d^{(k-1)T}Gd^{(k-1)}}{d^{(k-1)T}GD^{-1}Gd^{(k-1)}} & \text{otherwise}
\end{cases}
\]

\[(1.5)\]

1We recall that for \( \rho_k = 1, \theta_k = 1 \), for any \( k \), and \( 2D - G \) positive definite, we have the splitting method [26] (see also [18], [19]): for \( \rho_k = \rho < \frac{2}{\lambda_{\max}(D - \frac{1}{2}GD^{-1}G)} \), \( \theta_k = 1 \) for any \( k \), we have the projection method [30]; for \( \rho_k = \rho \), we have the scaled gradient projection method with a “limited minimization rule” [6].
\[ \rho_k = \begin{cases} 
\rho_{k-1} & \text{for } \|Gd^{(k-1)}\|^2 \leq \psi\|d^{(k-1)}\|^2 \\
\frac{d^{(k-1)T}Dd^{(k-1)}}{d^{(k-1)T}Gd^{(k-1)}} & \text{otherwise} 
\end{cases} \tag{1.6} \]

where \( \psi \) is a prefixed small tolerance.

An extensive numerical experimentation on large and sparse test problems has been carried out to show the effectiveness of the variable projection methods combined with the rules (1.5) and (1.6) (for a survey, see [41] and references therein). Nevertheless, the considered test problems are randomly generated with assigned features [11] or are chosen from the CUTE library [8].

The aim of this work is to show the numerical behaviour of the variable projection methods on a set of applications arising in the framework of data analysis.

In particular, in the next section, we describe the application of the VPM to the quadratic program arising in training the learning machines named Support Vector Machines (SVMs) [9]. In this QP problem the matrix \( G \) is dense and the feasible region is defined by box constraints and a single equality constraint. In the special instance of SVMs with Gaussian kernels, by introducing an appropriate updating rule for the projection parameters \( \rho_k \), a good convergence rate of the VPM is obtained even on this dense problem.

The second application concerns with the quadratic program arising in the numerical solution of an image restoration problem with a point–spread–function without a specific form [1]. The feasible region is defined as in the previous problem but the matrix \( G \) is sparse with a particular structure that is well exploited by the VPM.

The application described in the final section is related to a constrained bivariate interpolation problem [16]. A well known approach to this problem requires to solve a quadratic program with sparse block matrix \( G \) and sparse constraint matrices \( A \) and \( C \). In this case, the special definition of the entries of \( G \) suggests a very efficient splitting method for solving the QP problem. The effectiveness of the VPM is evaluated by comparison with this suitable splitting method.

2. Large QP Problems in Training Support Vector Machines

The Support Vector Machine is a technique for solving pattern recognition problems [7],[9],[36],[50]. Given a training set of labeled examples,

\[ D = \{(p_i, y_i), \ i = 1, \ldots, n, \ p_i \in \mathbb{R}^m, \ y_i \in \{-1, 1\}\}, \]
the SVM learning technique performs pattern recognition by finding a decision surface, \( F : \mathbb{R}^m \rightarrow \{-1, 1\} \), obtained by solving a quadratic program of the form:

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} x^T G x - \sum_{i=1}^{n} x_i \\
\text{subject to} & \quad \sum_{i=1}^{n} y_i x_i = 0, \quad 0 \leq x_j \leq C, \quad j = 1, \ldots, n,
\end{align*}
\]

(1.7)

where \( G \) is a symmetric positive semidefinite matrix related to the chosen decision surface. Since the matrix \( G \) is dense with size equal to the number of training examples, the problem is challenging \((n \gg 10000)\) in many interesting applications of the SVMs.

Among the various approaches proposed to overcome the difficulties involved in this problem, we may recognize two main classes. The first class collects the techniques based on different formulations of the optimization problem that gives rise to the separating surface (see [15] and references therein). These reformulations lead to more treatable optimization problems, but use criteria for determining the decision surface which, in some cases, are considerably different respect to the one of the standard SVM. The second class includes algorithms that exploit the special structure of problem (1.7) [14], [23], [37], [39]. In particular, the approaches proposed in [23], [37], [39] are based on special decomposition techniques that avoid explicit storage of \( G \) by splitting the original problem into a sequence of smaller QP subproblems of the form (1.7) with Hessian matrices equal to principal submatrices of \( G \). These techniques differ in the strategy employed for identifying the variables to update at each iteration and in the size chosen for the subproblems. In [39] the subproblems have size 2 and can be solved analytically, while in [23], [37] the subproblem size is a parameter of the procedure and a numerical QP solver is required.

In order to explain how the VPM can be a suitable solver for these QP subproblems we briefly sketch the SVM technique. Starting from the case of linear decision surface and linearly separable examples, the goal is to determine the hyperplane that leaves all the examples with the same label on the same side and maximizes the margin between the two classes, where the margin is defined as the sum of the distances of the hyperplane from the closest point of the two classes. Thus [9], the linear classifier

\[
F(p) = \text{sign} \left( w^* p + b^* \right), \quad w^* \in \mathbb{R}^m, \quad b^* \in \mathbb{R},
\]

is obtained from the solution \((w^*, b^*)\) of the following problem

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} w^T w \\
\text{subject to} & \quad y_i (w^* p_i + b) \geq 1 \quad i = 1, \ldots, n
\end{align*}
\]

(1.8)
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 [40]. In this case, the linear classifier comes from the solution $(\mathbf{w}^*, b^*, \xi^*)$ of

$$
\begin{align*}
\text{minimize} & \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \\
\text{subject to} & \quad y_i (\mathbf{w}^T \mathbf{p}_i + b) \geq 1 - \xi_i \quad i = 1, \ldots, n \quad (1.9)
\end{align*}
$$

The pair $(\mathbf{w}^*, b^*)$ can be obtained by applying a QP solver to (1.9) (or to (1.8) in the separable case); nevertheless, in order to generalize the procedure to nonlinear decision surfaces, it is useful to look at the dual problem. The dual problem of (1.9) is a convex QP problem of the form (1.7) with $G_{ij} = y_i y_j \mathbf{p}_i^T \mathbf{p}_j$ (the dual of (1.8) differs from (1.7) for the absence of the upper bound on the variables). If $\mathbf{x}^*$ denotes the solution of the dual problem, by using the KKT conditions for the primal problem, we may express $(\mathbf{w}^*, b^*)$ as

$$
\begin{align*}
\mathbf{w}^* &= \sum_{i=1}^n x_i^* y_i \mathbf{p}_i, \\
b^* &= y_j - \mathbf{w}^{*T} \mathbf{p}_j \quad \text{for any } j \text{ such that } 0 < x_j < C.
\end{align*}
$$

Thus, the classifier can be written as a linear combination of the training vectors associated to the nonzero multipliers. These training vectors, named Support Vectors (SVs), include all the information contained in the training set which is needed 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 $\mathbf{p} \in \mathbb{R}^m$ is an input point, let $\varphi(\mathbf{p})$ be the corresponding feature point with $\varphi$ a mapping from $\mathbb{R}^m$ to $Z$. The solution to the classification problem in the feature space will have the following form

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

and therefore it 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 (1.7) where

$$
G_{ij} = y_i y_j \varphi(\mathbf{p}_i)^T \varphi(\mathbf{p}_j).
$$
At first sight it might seem that the nonlinear separating surface can not be determined unless the mapping $\varphi$ is completely known. Nevertheless, since $\varphi$ appears only in scalar products between two feature points, if we find an expression for the scalar product in feature space which uses the points in input space only, that is

$$\varphi(p_i)^T \varphi(p_j) = K(p_i, p_j),$$

then, it is not necessary a full knowledge of the function $\varphi$. The symmetric function $K$ in (1.10) 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 (1.10). Two frequently used kernels are the polynomial kernel, $K(p_i, p_j) = (1 + p_i^T p_j)^d$, $d \in N\{0\}$, and the Gaussian kernel, $K(p_i, p_j) = exp(-\|p_i - p_j\|^2/(2\sigma^2))$, $\sigma \in R$. 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. Thus, from the computational point of view, the training of a SVM in the nonlinear case requires to solve a problem of form (1.7) where the entries of $G$ are defined by special kernel functions. In case of Gaussian SVMs, an example of the pattern of $G$ is shown in Figure 1.1. We may observe that the matrix is dense but with some particular features: the main diagonal has entries equal to 1, the other entries are in $[−1, 1]$ and many of them are much less than 1 in absolute value. These considerations suggest that the VPM, with $D$ diagonal matrix, may be an iterative solver suited for exploiting both the structure of the constraints and the particular nature that the Hessian matrix presents in the case of Gaussian SVMs. In fact, if $D$ is a diagonal matrix, each VPM subproblem is a separable QP problem subject to a single
linear equality constraint and box constraints, that is, a nonexpensive task that may be faced by efficient method [32], [38]. Furthermore, in [51] it is shown that, for the QP problem arising in training Gaussian SVMs, a good convergence rate of the VPM is obtained by using the following updating rule for the projection parameter $\rho_{k+1}$:

$$
\rho_{k+1} = \begin{cases}
\rho_k & \text{if } \|Gd^{(k)}\|^2 \leq \epsilon\|d^{(k)}\|^2 \\
\frac{d^{(k)T}Gd^{(k)}}{d^{(k)T}GS^{-1}Gd^{(k)}} & \text{if } \text{mod}(k, \tilde{k}) < \frac{\tilde{k}}{2} \\
\frac{d^{(k)T}Sd^{(k)}}{d^{(k)T}Gd^{(k)}} & \text{otherwise}
\end{cases}
\tag{1.11}
$$

It is interesting to observe that, on this dense problem, an effective updating rule for $\rho_{k+1}$ is obtained by combining the two rule (1.5) and (1.6) that, on sparse problems, exhibit good performance even if singly used. Unfortunately, in case of polynomial kernels, since the different definition of the dense Hessian matrix, the proposed updating rule is not so effective.

In the following, we report the results of a comparison between the VPM and the solver pr_LOQO actually used as inner QP solver in the decomposition scheme implemented in the package SVMlight [23]. The pr_LOQO is a version of the primal-dual infeasible interior point method of Vanderbei [49], appropriately implemented by Smola [46] for the dense problems of form (1.7). We consider small to medium test problems; of course, for large scale problems the two methods may be used as inner solvers of the decomposition techniques. The test problems are obtained by training Gaussian SVMs on the MNIST database of handwritten digits from AT&T Research Labs [27] and on the UCI Adult data set [31]. We construct test problems of size $N = 200, 400, 800, 1600$, by considering the first $N/2$ inputs of the digits 8 and 9. The UCI Adult data set allows to train a SVM to predict whether a household has an income greater than $50000. We consider the versions of the data set with size $N = 1605, 2265, 3185$. We use $C = 10, \sigma = 1800$ for the MNIST data set and $C = 1, \sigma^2 = 10$ for the UCI Adult data set.

The results in Tables 1.1 and 1.2 are obtained on a workstation Compaq XP1000 at 667MHz with 1GB of RAM, with code written in standard C. In the VPM, the matrix $D$ is chosen equal to the identity matrix, the stopping rule consists in the fulfillment of the KKT conditions within a tolerance of 0.001 (in general, a higher accuracy does not improve significantly the SVM performance [23],[39]) and for the parameter $\rho_k$ we use $\rho_1 = 1, \epsilon = 10^{-14}$ and $\tilde{k} = 6$ in the updating rule (1.11).
Table 1.1. Test problems from the MNIST data set

<table>
<thead>
<tr>
<th></th>
<th>n</th>
<th>iterations</th>
<th>time</th>
<th>SV</th>
<th>BSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>pr_LOQO</td>
<td>200</td>
<td>13</td>
<td>0.15</td>
<td>76</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>14</td>
<td>1.2</td>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>800</td>
<td>15</td>
<td>10.0</td>
<td>176</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1600</td>
<td>16</td>
<td>111.5</td>
<td>239</td>
<td>0</td>
</tr>
<tr>
<td>VPM</td>
<td>200</td>
<td>57</td>
<td>0.02</td>
<td>76</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>82</td>
<td>0.1</td>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>800</td>
<td>124</td>
<td>0.3</td>
<td>176</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1600</td>
<td>232</td>
<td>1.7</td>
<td>238</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1.2. Test problems from the UCI Adult data set

<table>
<thead>
<tr>
<th></th>
<th>n</th>
<th>iterations</th>
<th>time</th>
<th>SV</th>
<th>BSV</th>
</tr>
</thead>
<tbody>
<tr>
<td>pr_LOQO</td>
<td>1605</td>
<td>15</td>
<td>131.6</td>
<td>691</td>
<td>584</td>
</tr>
<tr>
<td></td>
<td>2265</td>
<td>15</td>
<td>383.9</td>
<td>1011</td>
<td>847</td>
</tr>
<tr>
<td></td>
<td>3185</td>
<td>15</td>
<td>1081.4</td>
<td>1300</td>
<td>1109</td>
</tr>
<tr>
<td>VPM</td>
<td>1605</td>
<td>136</td>
<td>2.3</td>
<td>691</td>
<td>584</td>
</tr>
<tr>
<td></td>
<td>2265</td>
<td>171</td>
<td>5.8</td>
<td>1011</td>
<td>847</td>
</tr>
<tr>
<td></td>
<td>3185</td>
<td>237</td>
<td>14.6</td>
<td>1299</td>
<td>1113</td>
</tr>
</tbody>
</table>

pr_LOQO solver was run with the parameter sigfig.max = 8 (a lower value implies an accuracy not comparable with that of the VPM). In the tables we report the number of iterations, the computational time in seconds required by the solvers, the number of support vectors and bounded support vectors (BSV), i.e. support vectors with \( x_i^* = C \).

From Tables 1.1 and 1.2 we may observe the higher effectiveness of the VPM especially when the size of the problem increases. In particular, the VPM allows the solution of medium size problems \((N > 1000)\) in few seconds. Thus, this method may be an useful inner QP solver in the decomposition techniques for very large-scale SVMs [51]. Finally, we emphasize that the method is easily parallelizable, since, in this application, each iteration consists essentially in a matrix-vector product and in the solution of a nonexpensive separable QP subproblem (eventually solvable in parallel [32]): hence, new parallel decomposition schemes can be based on the VPM [52].

3. Numerical Solution of Image Restoration Problem

The image degradation process can be described as a system which operates on an input image (object) \( \phi(\xi, \eta) \) to produce an output image (degraded image) \( \gamma(u, v) \). An additive noise term \( \epsilon(u, v) \) is also included.
Thus, the problem can be formulated as a first kind Fredholm integral equation:

\[
\gamma(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(u, \xi, v, \eta) \phi(\xi, \eta) d\xi d\eta + \epsilon(u, v) \quad (1.12)
\]

Additional constraints must hold; first the radiant energy distributions for input and output images must be positive or zero, i.e.

\[
\phi(\xi, \eta) \geq 0 \quad \text{for any } (\xi, \eta) \quad (1.13)
\]

\[
\gamma(u, v) \geq 0 \quad \text{for any } (u, v) \quad (1.14)
\]

Second, according to an energy conservation concept, the total energy in the object is preserved in the image, that is

\[
\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \phi(\xi, \eta) d\xi d\eta = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \gamma(u, v) dudv \quad (1.15)
\]

The problem of image restoration is the determination of the original object distribution \(\phi\) given the image \(\gamma\), the point–spread degradation function \(h\) and the noise \(\epsilon\). If we suppose that the function \(\phi, \gamma, h, \epsilon\) are sampled uniformly over a finite rectangular grid to form arrays of size \(n = \mu \times \nu\) (\(\mu\) and \(\nu\) are the numbers of inner grid points in \(u\) and \(v\) directions respectively) and use the rectangular rule for the approximation of the integral in (1.12), we obtain the following system of linear equations:

\[
\gamma = H\phi + \epsilon \quad (1.16)
\]

Here, \(\phi\) is an \(n\)–vector such that

\[
0 \leq \phi_i \leq \Phi \quad i = 1, \ldots, n \quad (1.17)
\]

where \(\Phi\) is the maximum gray level that we consider, and, from (1.15),

\[
\sum_{i=1}^{n} \phi_i = \sum_{i=1}^{n} \gamma_i \quad (1.18)
\]

The image restoration problem is a ill-conditioned problem (see [1]). A well known method for dealing with such problem is the method of regularization by Phillips and Tikhonov [48], that consists in solving the following minimization problem [1]

\[
\minimize \|H\phi - \gamma\|^2 + \lambda\|\Lambda\phi\|^2 \quad (1.19)
\]

where the matrix \(\Lambda\) is the regularization operator (usually chosen equal to the discretization of the Laplace or biharmonic operator) and \(\lambda\) is
the regularization parameter \((\lambda > 0); \) in the subsequent experiments \(\lambda = 10^{-4}\).

In many image restoration problems, the point–spread function \(h\) has the following general form, i.e., \(h\) is "space–variant" and "non–separable":

\[
h(u, \xi, v, \eta) = \frac{N}{2\pi \sigma(u, v)} \exp \left( -\frac{(\xi - u)^2 + (\eta - v)^2}{2\sigma^2(u, v)} \right) \tag{1.20}
\]

where \(N\) is a normalization constant for \(h\) and

\[
\sigma^2(u, v) = \begin{cases} 
    l\sin(u + v) & \text{if } \sin(u + v) \neq 0 \quad l \in [1, 2] \\
    l & \text{otherwise}
\end{cases} \tag{1.21}
\]

or

\[
\sigma^2(u, v) = l|u + v| \quad l \in [0.01, 0.02] \tag{1.22}
\]

Here, \(l\) is a positive constant that determines the extent of blur in the image; indeed for increasing values of \(l\), each image point is obtained with the contribution of a larger number of object points (superposition principle).

As for the application in the previous section, the VPM appears convenient for solving the QP problem (1.19) with the single equality constraint (1.18) and the simple bounds (1.17). In this case, the objective matrix \(G = H^TH + \lambda \Lambda^T \Lambda\) is sparse with a special structure; for example, when \(n = 6400\) and \(\sigma^2(u, v)\) as in (1.22), the pattern of the nonzero entries of \(G\) is showed in Figure 1.2 (here \(nz\) denotes the number of nonzero entries of \(G\)). Figure 1.3 is a detail of the previous pattern for the leading principal submatrix of order 320 of \(G\) while Figure 1.4 displays in 3–D the entries of this submatrix.

From practical point of view, it is convenient to store in compressed form only the nonzero elements of \(H\) and to avoid the explicit computation of \(H^TH\) in \(G\); the matrix–vector multiplication \(Gy(k)\) required by the VPM, may be computed by routines that exploit the sparsity of \(H\).

In this implementation of the VPM we use the identity matrix as \(D\), the updating rule (1.5) for \(\{\rho_k\}\) and the solver in [38] for the inner subproblems.

The three images in Figure 1.5 (see also Table 1.3) have been used to generate the test problems for evaluating the effectiveness of this approach.

Figure 1.6 shows four blurred images and Table 1.4 reports the parameters used in the related point–spread function and the three picture
Figure 1.2. Pattern of the nonzero entries of $G$

Figure 1.3. Detail of the pattern of the nonzero entries of $G$

Figure 1.4. 3-D pattern of the detail of $G$
Figure 1.5. Original images for the test problems

Table 1.3. Features of the original images

<table>
<thead>
<tr>
<th>image</th>
<th>size</th>
<th>gray level</th>
</tr>
</thead>
<tbody>
<tr>
<td>IM1</td>
<td>80 × 80</td>
<td>64</td>
</tr>
<tr>
<td>IM2</td>
<td>128 × 128</td>
<td>256</td>
</tr>
<tr>
<td>IM3</td>
<td>128 × 128</td>
<td>256</td>
</tr>
</tbody>
</table>

distance measures $D$, $R$, $E$, defined by Herman in [22]:

$$D = \frac{\sum_{\xi=1}^{\mu} \sum_{\eta=1}^{\nu} |\phi(\xi, \eta) - \gamma(\xi, \eta)|}{\sum_{\xi=1}^{\mu} \sum_{\eta=1}^{\nu} |\phi(\xi, \eta) - \tau|}$$

$$R = \frac{\sum_{\xi=1}^{\mu} \sum_{\eta=1}^{\nu} |\phi(\xi, \eta) - \gamma(\xi, \eta)|}{\sum_{\xi=1}^{\mu} \sum_{\eta=1}^{\nu} \phi(\xi, \eta)}$$

$$E = \max_{\xi, \eta} |\phi(\xi, \eta) - \gamma(\xi, \eta)|$$
where $\gamma(\xi, \eta)$ is the blurred image and $\tau$ denotes the average density of the digitized test image.

Table 1.4. Features of the blurred images

<table>
<thead>
<tr>
<th>test problem</th>
<th>original image</th>
<th>formula for $\sigma^2$</th>
<th>$D$</th>
<th>$R$</th>
<th>$E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP1</td>
<td>IM1</td>
<td>(1.22) $l = 0.01$</td>
<td>0.46</td>
<td>0.19</td>
<td>33</td>
</tr>
<tr>
<td>TP2</td>
<td>IM2</td>
<td>(1.21) $l = 1.3$</td>
<td>0.37</td>
<td>0.28</td>
<td>143</td>
</tr>
<tr>
<td>TP3</td>
<td>IM3</td>
<td>(1.22) $l = 0.01$</td>
<td>0.23</td>
<td>0.14</td>
<td>110</td>
</tr>
<tr>
<td>TP4</td>
<td>IM3</td>
<td>(1.21) $l = 1.5$</td>
<td>0.43</td>
<td>0.28</td>
<td>155</td>
</tr>
</tbody>
</table>

Figure 1.6. Blurred images for the test problems

Figure 1.7 shows the restored images and Table 1.5 summarize the results obtained by the VPM method. In particular $it$ denotes the number of required iterations to satisfy the Karush–Kuhn–Tucker conditions within a tolerance $10^{-3}$, while $time$ is the elapsed time expressed in seconds required for the restoration on a Compaq XP1000 workstation. $D, R$
and $E$ are the picture distance measures between the original image $\phi(\xi, \eta)$ and the restored image $\gamma(\xi, \eta)$.

Table 1.5. Features of the restored images

<table>
<thead>
<tr>
<th>test problem</th>
<th>it</th>
<th>time</th>
<th>$D$</th>
<th>$R$</th>
<th>$E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP1</td>
<td>101</td>
<td>35.0</td>
<td>0.21</td>
<td>$7.44 \cdot 10^{-2}$</td>
<td>18</td>
</tr>
<tr>
<td>TP2</td>
<td>138</td>
<td>135.1</td>
<td>$7.15 \cdot 10^{-2}$</td>
<td>$2.98 \cdot 10^{-2}$</td>
<td>124</td>
</tr>
<tr>
<td>TP3</td>
<td>131</td>
<td>245.8</td>
<td>0.11</td>
<td>$5.68 \cdot 10^{-2}$</td>
<td>94</td>
</tr>
<tr>
<td>TP4</td>
<td>169</td>
<td>193.9</td>
<td>0.13</td>
<td>$5.79 \cdot 10^{-2}$</td>
<td>115</td>
</tr>
</tbody>
</table>

4. A Bivariate Interpolation Problem

A bivariate interpolation problem can be stated as follows: given $N$ distinct and arbitrarily spaced points $V_i$, $i = 1, \ldots, N$ in a convex domain $\Omega$ in the $x$–$y$ plane and $N$ real numbers $z_i$, consider the problem of computing a $C^1$ bivariate function $F(x, y)$ whose values $V_i$ are exactly $z_i$. This problem can easily have thousands of interpolating points $V_i$. 
When some specific information is available for computing $C^1$ interpolants which are shape–preserving or when step gradients are implied by the data or values of $F(x, y)$ are rapidly varying in some subset of the domain $\Omega$, appropriate equality and inequality constraints are imposed on the first and second order partial derivatives of $F(x, y)$ on a given set of points. In this case we have a bivariate interpolation problem with constraints.

A general method for interpolation of scattered data can be described by three steps. **Triangulation**: divide the domain into subdomains by connecting the points $V_i$ appropriately ([25], [2]). **Curve network**: compute a smooth function on the boundary of each subdomain (e.g. see [33], [34]). **Blending**: construct the patch that interpolates the functions and their derivatives over each subdomain and fit the patches together to form a $C^1$ surface [4], [5], [24]).

The second step has a high computational complexity: indeed, one must determine the first order partial derivatives of $F(x, y)$ in the interpolation points $V_i$, $i = 1, \ldots, N$, and define the constraints on the $C^1$ surface $z = F(x, y)$. That is, one must solve a convex quadratic programming problem, which, generally, has more than a hundred and perhaps thousands of variables and constraints.

If $F_x(V_i)$ and $F_y(V_i)$ denote the first order partial derivatives of $F(x, y)$ in $V_i$ and $u$ is the $2N$–vector

$$u = (F_x(V_1), F_y(V_1), F_x(V_2), F_y(V_2), \ldots, F_x(V_N), F_y(V_N))^T$$

we can express the value of $F(x, y)$ in a point $Q$ in the triangle of vertices $V_i, V_j$ and $V_k$ as a linear combination of $F(V_i), F(V_j), F(V_k), F_x(V_i), F_y(V_i), F_x(V_j), F_y(V_j), F_x(V_k)$ and $F_y(V_k)$.

The first and the second order partial derivatives of $F(x, y)$ in $Q$ are expressed analogously. Thus the constraints on the surface $F(x, y)$ can be expressed as a set of linear equalities and inequalities of the form

$$c_i^T u = d_i, \quad i = 1, \ldots, m_e;$$

$$a_i^T u \geq b_i, \quad i = 1, \ldots, m_i;$$

(1.23)

where $m_e$ and $m_i$ are the number of equality and inequality constraints respectively.

Nielsen in [33] suggests a method to compute partial derivatives based upon a minimum norm network. This method consists in computing a set of cubic Hermite polynomial on the domain $E$, union of all the edges of the triangulation of $\Omega$, which interpolate $z_i$ in $V_i$, $i = 1, \ldots, N$ and in computing first order partial derivatives by minimizing an appropriate functional on $E$. 


Analogous to the minimum pseudonorm property of univariate splines, the proposed quadratic functional is

\[ J(F) = \sum_{ij \in N_e} \int_{e_{ij}} \left( \frac{\partial^2 F}{\partial e_{ij}^2} \right)^2 d\xi_{ij} \]  

(1.24)

where \( d\xi_{ij} \) represents the arc length along the edge \( e_{ij} \) in the triangulation with endpoints \( V_i \) and \( V_j \) and \( N_e \) is a list of indices representing the edges of the triangulation. Let \( C[E] \) be the set of all \( C^1 \) functions \( F \) defined on \( \Omega \) restricted to \( E \), such that the univariate function \( f \), obtained as restriction of \( F \) on \( e_{ij} \), is an element of \( H[e_{ij}] \), with \( H[e_{ij}] = \{ f : f \in C[e_{ij}], f' \text{ absolutely continuous}, f'' \in L^2[e_{ij}] \} \).

It is convenient to view each \( F \in C[E] \) as a collection of univariate functions \( f_{ij}(\tau) = F((1 - \tau)V_i + \tau V_j) \) \( i, j \in N_e \) \( 0 \leq \tau \leq 1 \)

and, thus, the problem of minimizing the functional (1.24) can be written as the following quadratic programming problem:

\[ \min \frac{1}{2} u^T Gu^T + u^T q \]  

(1.25)

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

\[ G_{ii} = \begin{pmatrix} \tilde{a}_i & \tilde{c}_i \\ \tilde{c}_i & \tilde{b}_i \end{pmatrix} \quad G_{ij} = \begin{cases} \begin{pmatrix} \tilde{a}_{ij} & \tilde{c}_{ij} \\ \tilde{c}_{ij} & \tilde{b}_{ij} \end{pmatrix} & \text{if } i, j \in N_i \\ 0 & \text{otherwise} \end{cases} \]  

(1.26)

with

\[ \tilde{a}_{ij} = \frac{(x_j - x_i)^2}{2||e_{ij}||^3} \quad \tilde{c}_{ij} = \frac{(y_j - y_i)(z_j - z_i)}{2||e_{ij}||^3} \]

\[ \tilde{b}_{ij} = \frac{(y_j - y_i)^2}{2||e_{ij}||^3} \]

and

\[ q_i = \begin{pmatrix} -\frac{3}{2} \sum_{ij \in N_i} \frac{(x_j - x_i)}{||e_{ij}||^3}(z_j - z_i) \\ -\frac{3}{2} \sum_{ij \in N_i} \frac{(y_j - y_i)}{||e_{ij}||^3}(z_j - z_i) \end{pmatrix} \]
In Figure 1.8 it is showed the pattern of the nonzero entries (denoted as \( nz \)) of the matrix \( G \) with \( N = 1000 \); in Figure 1.9 a pattern of the
nonzero entries of a principal submatrix of $G$ of order 100 is displayed; the entries of this submatrix are also displayed in 3–D in Figure 1.10. It is possible to prove (33) that, among all functions $F \in C[E]$, $F(V_i) = z_i$, $i = 1, ..., N$, the function $S \in C[E]$ that uniquely minimizes $J(F)$ in (1.24) is a set of univariate cubic Hermite interpolants $(i, j \in N_e, 0 \leq \tau \leq 1)$

$$s_{ij}(\tau) = S((1 - \tau)V_i + \tau V_j)$$

$$= \tau^2(3 - 2\tau)z_j + (1 - \tau)^2(2\tau + 1)z_i +$$

$$+ \tau(1 - \tau)^2[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i)] +$$

$$+ \tau^2(\tau - 1)[(x_j - x_i)S_x(V_j) + (y_j - y_i)S_y(V_j)]$$

Thus, the vector $u^* = (u_1^*, ..., u_N^*^T)^T$, with $u_i^* = (S_x(V_i), S_y(V_i))^T$, $i = 1, ..., N$, is the solution of the quadratic problem (1.25)–(1.23).

In [16] it is proved that the symmetric matrix $G$ is positive definite. If we set $D = \text{diag}(G_{11}, ..., G_{NN})$ and $H = G - D$, in [18] it is proved that the symmetric matrix $D - H$ is positive definite (i.e. $(D, H)$ is a P–regular splitting [35, p. 122]), thus splitting method for the strictly convex quadratic problem (1.25)–(1.23) can be applied (see [18], [19]). Sometimes, it is convenient to compute the estimates of the first partial order derivatives by minimizing, in the class of exponential Hermite polynomials ([34]) or in the class of cubic Hermite polynomials ([3]) the tensioned quadratic functional

$$J_\alpha(F) = \sum_{ij \in N_e} \left[ \int_{e_{ij}} \left( \frac{\partial^2 F}{\partial e_{ij}^2} \right)^2 d\xi_{ij} + (\alpha_{ij})^2 \int_{e_{ij}} \left( \frac{\partial F}{\partial e_{ij}} \right)^2 d\xi_{ij} \right]$$

where $\alpha_{ij}$ are nonnegative tension parameters, that can assume different values at each edge of the triangulation $e_{ij}$. In general it is used a normalized tension parameter $\alpha = \alpha_{ij}\|e_{ij}\|$. In both the cases (cubic or exponential Hermite polynomials) the Hessian matrices have the same block structure of (1.26). In [17] it is proved that, in the case of exponential Hermite polynomials, the Hessian matrix is symmetric positive definite and a P–regular splitting can be derived for the functional (1.24) by the $2 \times 2$ block diagonal part of the Hessian matrix. In [17], it is also proved that, in the case of cubic Hermite polynomial, the Hessian matrix is block strictly (or block irreducibly) diagonally dominant (see [13]).
In order to solve problem (1.25)–(1.23) by splitting method, we have to solve at each iteration a convex quadratic subproblem

\[
\begin{aligned}
\text{minimize} & \quad \frac{1}{2} u^T D u + (q + H u^{(k-1)})^T u \\
\text{subject to} & \quad C u = d, \quad A u \geq b,
\end{aligned}
\tag{1.28}
\]

where

\[
C = \begin{bmatrix}
  c_1^T \\
  \vdots \\
  c_m^T
\end{bmatrix}, \quad A = \begin{bmatrix}
  a_1^T \\
  \vdots \\
  a_m^T
\end{bmatrix}
\]

This subproblem can be formulated as a mixed linear complementarity problem

\[
\begin{pmatrix}
  s \\
  0
\end{pmatrix} =
\begin{pmatrix}
  z_1^{(k)} \\
  z_2^{(k)}
\end{pmatrix} +
M \begin{pmatrix}
  \lambda \\
  \mu
\end{pmatrix}
\quad (1.29)
\]

where

\[
z_1^{(k)} = -b - AD^{-1}(q + H u^{(k-1)});
\quad z_2^{(k)} = -d - CD^{-1}(q + H u^{(k-1)})
\]

and the symmetric positive semidefinite matrix \(M\) of order \(m_i + m_e\) is given by

\[
M = \begin{pmatrix}
  AD^{-1} A^T & AD^{-1} C^T \\
  CD^{-1} A^T & CD^{-1} C^T
\end{pmatrix}
\]

If

\[
\begin{pmatrix}
  \lambda^{(k)} \\
  \mu^{(k)}
\end{pmatrix}
\]

denotes the solution of (1.29), the solution of (1.28) can be computed by

\[
u^{(k)} = -D^{-1} (q + H u^{(k-1)} - A^T \lambda^{(k)} - C^T \mu^{(k)})
\]

We observe that the matrix \(CD^{-1} C^T\) is a symmetric positive definite matrix.

In the case of equality constraints only, splitting method is examined in [12]; the mixed LCP (1.29) is reduced to a symmetric positive definite linear system that can be solved by conjugate gradient method.

In order to solve the large-scale mixed LCP (1.29) (such as the one which arises from the bivariate interpolation problem), iterative schemes such as Cryer’s SOR ([10]), are very efficient.

We can also consider parallel algorithms for LCP such as Parallel-SOR ([28]), gradient projection–SOR ([29]) and overlapping parallel-SOR ([20]).
A deep analysis on the evaluation of the performances of the parallel algorithms above on a distributed memory system (such as Cray MPP T3E at the CINECA Supercomputing Center in Bologna), has been developed in [21] for the mixed LCP which arises from the bivariate interpolation problem. In the numerical experiments in Table 1.6, we report the behaviour of the splitting method applied to the bivariate interpolation problem by solving the inner mixed LCP by Cryer’s SOR. The stopping rule for the iterations of the splitting method (outer iterations) is that the Karush–Kuhn–Tucker optimality conditions of the QP problem (1.25)–(1.23) hold within the tolerance $10^{-9}$

$$\|G u^{(k)} + q - A^T \lambda^{(k)} - C^T \mu^{(k)}\|_\infty \leq 10^{-9}$$

The stopping rule for the iterations of the inner solver (inner iterations) is an adaptive criteria related to how the current approximation of the solution and the associate multipliers satisfy the KKT optimality conditions.

The computational experiments have been carried out on a Cray C90 (machine precision $\approx 7 \cdot 10^{-15}$).

<table>
<thead>
<tr>
<th>Table 1.6. Behaviour of Splitting Method</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>$n$</th>
<th>$m_e$</th>
<th>$m_i$</th>
<th>outer iterations</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>200</td>
<td>300</td>
<td>162</td>
<td>0.743</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>500</td>
<td>133</td>
<td>1.68</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>1000</td>
<td>113</td>
<td>5.94</td>
</tr>
<tr>
<td>5000</td>
<td>500</td>
<td>500</td>
<td>167</td>
<td>20.3</td>
</tr>
<tr>
<td></td>
<td>2500</td>
<td>2500</td>
<td>125</td>
<td>24.7</td>
</tr>
<tr>
<td>10000</td>
<td>3000</td>
<td>4000</td>
<td>135</td>
<td>41.0</td>
</tr>
</tbody>
</table>

In Table 1.7, it is reported a comparison between the splitting method and the variable projection method (both with Cryer’s SOR as inner solver) for the solution of a bivariate interpolation problem of size $n = 3000$. The comparison between the two methods has been done in terms of outer iterates, total amount of inner iterates (reported in brackets) and elapsed time expressed in seconds.
The sparsity of the matrices $G$ and $(CTA^T)$ are 99.6% and 99.7% respectively; the condition number of the matrix $G$ is $K(G) = 146.5$.

Here the stopping rule for both the methods is:

$$\frac{\|u^{(k+1)} - u^{(k)}\|}{\|u^{(k+1)}\|} \leq 10^{-6}$$

The experiments have been carried out on a Digital–Compaq workstation 500au.

Table 1.7. Comparison Splitting Method – Variable Projection Method

<table>
<thead>
<tr>
<th>Splitting Method</th>
<th>Variable Projection Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D = \text{diag}(G_{ii})$</td>
<td>$D = \text{diag}(G_{ii})$</td>
</tr>
<tr>
<td>$m_x$</td>
<td>$m_y$</td>
</tr>
<tr>
<td>200</td>
<td>100</td>
</tr>
<tr>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>1000</td>
<td>1000</td>
</tr>
</tbody>
</table>

5. Conclusions

The following conclusions can be drawn:

- The effectiveness of the variable projection method has been evaluated by numerical experiments on test problems from CUTE library and randomly generated (by varying size, conditioning and sparsity of Hessian and constraints matrices) in [41].

The comparison with other similar iterative schemes (classical splitting and projection methods, modified projection method ([47]), scaled gradient projection method) emphasized the definite advantage of VPM which uses a variable projection parameter $\rho_k$ at each iteration.

- The efficiency of all the above iterative schemes is strictly related with an efficient solver for the inner subproblem of each iteration (inner solver). This is strictly connected with the structure of the constraints or how to reformulate the inner subproblem.

- The applications in sections 2 and 3 lead to quadratic programs with special structure of constraints but different sparsity of Hessian matrices. Efficient direct solvers for the inner subproblem (quadratic knapsack problem) are available in literature.
The VPM solves efficiently the QP problem of Section 3 (Hessian matrix is very sparse) and it seems to be attractive also for the solution of the problem of Section 2, despite the density of the Hessian matrix.

In the application of Section 4, the constraints are general and the Hessian matrix has a particular block structure. The inner subproblem is reformulated as a mixed linear complementarity problem; the inner iterative solver is very efficient because of the sort of block diagonally dominance of the Hessian matrix which “positively influences” the complementarity matrix.

Because of this sort of block diagonally dominance of the Hessian matrix, there is no considerable advantage to use the variable projection method respect to the classical splitting method suggested by the special structure of the Hessian (see Table 1.7).
References


REFERENCES


