

QPECgen, a MATLAB generator for mathematical programs with quadratic objectives and affine variational inequality constraints

HOUYUAN JIANG

jiang@ms.unimelb.mu.oz.au

DANIEL RALPH

danny@ms.unimelb.mu.oz.au

*Department of Mathematics and Statistics
The University of Melbourne
Parkville, Vic. 3052, Australia*

Editor: Jong-Shi Pang

Abstract. We describe a technique for generating a special class, called QPEC, of mathematical programs with equilibrium constraints, MPEC. A QPEC is a quadratic MPEC, that is an optimization problem whose objective function is quadratic, first-level constraints are linear, and second-level (equilibrium) constraints are given by a parametric affine variational inequality or one of its specialisations. The generator, written in MATLAB, allows the user to control different properties of the QPEC and its solution. Options include the proportion of degenerate constraints in both the first and second level, ill-conditioning, convexity of the objective, monotonicity and symmetry of the second-level problem, and so on. We believe these properties may substantially effect efficiency of existing methods for MPEC, and illustrate this numerically by applying several methods to generator test problems. Documentation and relevant codes can be found by visiting <http://www.maths.mu.OZ.AU/~danny/qpecgendoc.html>.

Keywords: Mathematical programs with equilibrium constraints, bilevel program, constrained optimization, quadratic program with affine variational inequality constraints, degeneracy, ill-conditioning, optimality conditions, variational inequality, complementarity problem, MATLAB, test problem generator.

A tribute to Olvi Mangasarian, on the occasion of his 65th birthday

I knew of the Mangasarian-Fromovitz constraint qualification for nonlinear programs before I started as a graduate student at the University of Wisconsin-Madison, so I was really excited to have Professor Mangasarian as an instructor in my first semester on campus. Olvi's teaching style was engaging and expansive. For example, he once posed a question to the class with the promise of a chocolate bar, "What is the best application for a nice method of finding the largest Euclidean ball within a polyhedron?" I remember that Olvi liked my suggestion — King Arthur wants to construct the largest round table that will fit in his irregular, polyhedral great hall — but I don't recall what happened to the chocolate! Of course, as I found later, warmth and contagious enthusiasm are two of Olvi's trademarks.

Although Olvi did not supervise my research studies, I bumped into his significant and often diverse contributions, like exact penalty analysis for nonlinear programs, iterative methods for solving complementarity problems, and determining whether cancer tumours are malignant or benign using linear programming. The breadth

and depth of his contributions are matched by, in fact are motivated by, his incredible stream of high quality research ideas and energy.

It is with fondness that I remember discussions with Olvi at the UW-Madison; and a sense of privilege that I have come to know him as a colleague.

Danny Ralph.

1. Introduction

Mathematical Programs with Equilibrium Constraints, MPEC, is an important and new field in mathematical programming with roots in game theory and bilevel optimization. It has found a number of significant applications in economics and engineering sciences. See [19] for applications and references; see also the survey [31] on the closely related area of bilevel programming.

Due to the complex structure of the feasible set of MPEC, the analysis of optimality conditions and the development of algorithms are associated with more difficulties than ordinary nonlinear programming problems. Nevertheless, much progress has been made in recent years. We will focus on several recently announced algorithms designed to find local minimizers of MPEC: the penalty interior-point algorithm (PIPA for short) [19], smoothing methods [9, 14], which are related to the interior-point approach, and piecewise sequential quadratic programming (PSQP) [19, 20, 28]. Penalty interior-point algorithms converge globally under some suitable conditions that include strict complementarity of the lower-level solution, while the piecewise SQP method exhibits local superlinear convergence under the uniqueness of multipliers and some second-order sufficient conditions, but without requiring a strict complementarity condition. Some preliminary numerical experiments have been carried out for PIPA and PSQP [19, 20], and smoothing methods [9, 14]. The theoretical results and numerical experience show some promise for these recent methods.

We mention, among others, implicit programming methods, exact penalty approaches and descent methods which will not be considered in this paper. Implicit programming reformulations of MPEC aim to remove the equilibrium constraints by solving for, or eliminating some variables in terms of others. The resulting problem is often a simply constrained optimization problem with a nonconvex and nonsmooth Lipschitz objective function for which bundle methods are available. The theoretical development and numerical experiments along this direction can be found in [7, 8, 10, 12, 19, 25, 26]. Using exact penalization of equilibrium constraints, MPEC can also be converted to nonsmooth optimization problems with smooth constraints; see [1, 3, 19, 21, 30]. Based on exact penalization, a trust region method is proposed in [30] and its global convergence is established. Several descent methods can be found from [16, 23, 32]. Note that the descent method proposed in [16] is similar to the aforementioned PSQP. For other numerical algorithms of MPEC, see [2, 19, 31].

Efficiency of MPEC methods is an important issue that has hardly been explored. The development of numerical methods of MPEC is still in the early stages and lags somewhat behind the theory [19]. More and more practical problems have been

studied. However, there is still a need for good test examples to judge numerical performance of existing methods.

The purpose of this paper is two-fold. The first goal is to randomly generate test problems for the QPEC, a mathematical program with a quadratic objective function, polyhedral “first-level” constraints, and “second-level” affine variational inequality constraints, while allowing user controls on certain properties of the problem and its solution. Properties of QPEC we believe are important include constraint degeneracy and the degree of degeneracy of both the first-level and the second-level constraints, conditioning of the objective function and the second-level problem, and convexity and monotonicity properties of the objective function and the second-level variational inequality respectively. We mention that degeneracy (the lack of strict complementarity) in the solution of the second-level variational inequality gives rise to potentially huge numbers of subproblems that need to be examined in order to check stationarity. The second goal of the paper is to compare MPEC methods which are designed to find local minimizers, using test problems provided by the generator. Some preliminary conclusions are drawn from the numerical results.

Algorithms designed to find global minimizers of MPEC are beyond the scope of this paper. Note that Calamai and Vicente [4] have implemented a generator of bilevel programs aimed at global optimization, which allows user controls on the number and type of local versus global minimizers. In the next section we give some examples of QPEC with multiple local minimizers and make some comparison with the generator of [4].

The paper is organized as follows. In the next section, the MPEC is formally defined and its reformulation via KKT conditions is presented. Section 3 is a brief discussion of optimality conditions of the MPEC which are the basis for developing the generator. We explain the main features and input data of the generator, and present the generator QPECgen in Section 4. Section 5 is very large compared with other sections, where we describe several algorithms we are going to test for the problems generated by QPECgen. Numerical results are also reported and discussed in this section. Finally, we conclude the paper by giving some discussions and possible future work.

2. Formulation

Consider an MPEC with affine variational inequality constraints:

$$\begin{aligned} & \underset{(x,y) \in \mathfrak{R}^{n+m}}{\text{minimize}} && f(x,y) \\ & \text{subject to} && (x,y) \in Z = \{(x,y) : Gx + Hy + a \leq 0\} \\ & && y \in \mathcal{S}(x) = \{y \in C(x) : \forall v \in C(x), (v - y)^T F(x,y) \geq 0\}, \end{aligned} \tag{1}$$

where $f : \mathfrak{R}^{n+m} \rightarrow \mathfrak{R}$ is twice continuously differentiable, $G \in \mathfrak{R}^{l \times n}$, $H \in \mathfrak{R}^{l \times m}$, $a \in \mathfrak{R}^l$,

$$\begin{aligned} F(x, y) &= Nx + My + q && \in \mathfrak{R}^m, \\ C(x) &= \{y : g(x, y) \leq 0\} && \in \mathfrak{R}^m, \\ g(x, y) &= Dx + Ey + b && \in \mathfrak{R}^p, \end{aligned}$$

and $N \in \mathfrak{R}^{m \times n}$, $M \in \mathfrak{R}^{m \times m}$, $q \in \mathfrak{R}^m$, $D \in \mathfrak{R}^{p \times n}$, $E \in \mathfrak{R}^{p \times m}$, $b \in \mathfrak{R}^p$. From (1) we see that $\mathcal{S}(x)$ is the solution set of an affine variational inequality, AVI, that is parametric in x .

We will mainly be interested in QPEC of the form (1) that is when

$$f(x, y) = \frac{1}{2}(x, y)^T P(x, y) + c^T x + d^T y,$$

where $P \in \mathfrak{R}^{(n+m) \times (n+m)}$, $c \in \mathfrak{R}^n$, $d \in \mathfrak{R}^m$.

A QPEC of the form (1) belongs to the class of mathematical programs with affine equilibrium constraints or MPAEC, also called AVI-constrained MP or AVI-MP since the equilibrium constraints specify an affine variational inequality. We now refer to a QPEC of this type as a QPAEC or AVI-QP. The condition $(x, y) \in Z$ is called a first-level or upper-level or outer constraint. The equilibrium condition $y \in \mathcal{S}(x)$ is called a second-level or lower-level or inner constraint. Note that in general, MPEC differ from bilevel programs in that the second-level constraint $y \in \mathcal{S}(x)$ represents a parametric (nonlinear) VI in the former, and a parametric nonlinear program in the latter; hence an MPEC is entirely equivalent to a bilevel program if the second-level VI is the optimality condition for a convex program.

Implicit programs are an important subclass of two-level optimization problems in which the first-level constraints only impinge on the first-level variables. Implicit QPEC is the class of problems of the type (1) or one of its variants below, with f quadratic, in which

$$Z = X \times \mathfrak{R}^m, \quad X = \{x : Gx + a \leq 0\}.$$

Some algorithms such as presented in [18, 25, 26] depend on the implicit program structure by attempting to eliminate y as a function of x , and solve a reduced-dimension problem with a generally nondifferentiable objective function $f(x, y(x))$ subject to constraints only on $x \in X$.

Suppose $x \in \mathfrak{R}^n$ is given. Note for the second-level AVI $y \in \mathcal{S}(x)$, that the function F is affine and the feasible region $C(x)$ is polyhedral. Thus the first-order necessary conditions, namely the Karush-Kuhn-Tucker (KKT) conditions, characterise solvability: y belongs to $\mathcal{S}(x)$ if and only if there exists a multiplier vector $\lambda \in \mathfrak{R}^p$ such that

$$\begin{aligned} F(x, y) + E^T \lambda &= 0 \\ g(x, y) &\leq 0, \quad \lambda \geq 0, \quad \lambda^T g(x, y) = 0. \end{aligned} \tag{2}$$

So the MPAEC can be reformulated as a KKT-constrained mathematical program or KKT-MP, which is also a nonlinear program (NLP):

$$\begin{aligned} &\text{minimize}_{(x, y, \lambda) \in \mathfrak{R}^{n+m+p}} && f(x, y) \\ &\text{subject to} && (x, y) \in Z \\ & && F(x, y) + E^T \lambda = 0 \\ & && g(x, y) \leq 0, \quad \lambda \geq 0, \quad \lambda^T g(x, y) = 0. \end{aligned} \tag{3}$$

By specifying D, E and b in g , we may obtain an important special case of (3) which is termed as MPEC with box constraints (Box-MP for short):

$$\begin{aligned}
& \underset{(x,y)}{\text{minimize}} && f(x,y) \\
& \text{subject to} && (x,y) \in Z \\
& && l_i < y_i < u_i \implies F_i(x,y) = 0 \\
& && y_i = l_i \implies F_i(x,y) \geq 0 \\
& && y_i = u_i \implies F_i(x,y) \leq 0,
\end{aligned} \tag{4}$$

where $-\infty \leq l_i < u_i \leq \infty$ for $i = 1, 2, \dots, m$. Furthermore, if $l_i = 0, u_i = \infty$ for $i = 1, 2, \dots, m$, then the above problem reduces to a mathematical program with linear complementarity constraints, or LCP-MP:

$$\begin{aligned}
& \underset{(x,y)}{\text{minimize}} && f(x,y) \\
& \text{subject to} && (x,y) \in Z \\
& && 0 \leq F(x,y) - y \geq 0,
\end{aligned} \tag{5}$$

where $-$ denotes orthogonality.

If f is quadratic, the problems (3), (4) and (5) may be called KKT-QP, Box-QP and LCP-QP respectively.

At first glance, the AVI-MP (3) is nothing but a nonlinear programming problem; indeed if f is quadratic, it would be a quadratic program except for the bilinear (nonlinear) constraint $\lambda^T g(x, y) = 0$. Similarly the equilibrium constraints of the Box-MP (4) and LCP-MP (5) can be written as bilinear equality constraints. Therefore it appears that MPEC can be studied and solved using nonlinear programming techniques. Unfortunately, a significant difficulty is that the usual constraint qualifications in nonlinear programming fail to be satisfied even when the MPEC has very fine properties such as strong monotonicity of the second-level function F with respect to the second-level variable y . This casts considerable doubt about the applicability of standard nonlinear programming approaches to MPEC. See [5, 19, 29] for more discussion.

The following simple examples help to explain why the MPEC is such a difficult problem in optimization.

Example 1. Let $f : \Re^2 \rightarrow \Re$ and the MPEC be defined by

$$\begin{aligned}
& \text{minimize} && f(x,y) = (x+1)^2 + (y-2)^2 \\
& \text{subject to} && 0 \leq (y-x) - y \geq 0.
\end{aligned}$$

Notice that there are no first-level constraints, that the objective function f is strictly convex with respect to (x, y) , and that, given x , the second-level problem is a strongly monotone linear complementarity problem in y . For any fixed $x \in \Re$, the second-level problem LCP has a unique solution. One may find that the feasible solution set of this LCP-QP is $\{(x, 0) : x \leq 0\} \cup \{(x, x) : x \geq 0\}$. It is interesting to note that this LCP-QP has two local minimizers $(-1, 0)$ and $(0.5, 0.5)$. This demonstrates the well-known fact that an MPEC may have more than one local minimizers in spite of rather nice properties of the lower-level VI.

Example 2. Suppose a linear inequality $2y - x - 1 \geq 0$ is added as a first-level constraint in Example 1. It is easy to see that the feasible solution set of the LCP-QP: $\{(x, 0) : x \leq -1\} \cup \{(x, x) : x \geq 1\}$ is unconnected. There is no doubt that this LCP-QP has more than one local minimizer.

Calamai and Vicente [4] developed a quadratic bilevel programming generator, the basic model of which has some similarity to Example 3 below. The objective function of the bilevel program discussed in [4] is a strictly convex quadratic function, and the second-level problem is a strictly convex quadratic program with bounded convex polyhedral constraints. It is known that a convex quadratic program is equivalent to a monotone affine variational inequality problem. Therefore, the test problems produced by Calamai and Vicente's generator are equivalent to the MPEC problem with a strictly convex objective function, and with the second level problem being an affine variational inequality problem. However, the second-level constraint set of Calamai and Vicente's test problem is bounded in contrast to unbounded ones (nonnegative orthants for any x) in Examples 3 and 4 which are presented below. Moreover, the parameters defining those bounds play an important role as far as the number of local and global minimizers is concerned. More importantly, Examples 3 and 4 are just some simple examples provided by our generator, which are not relevant to the main part of our generator. More complicated examples generated by Calamai and Vicente's generator heavily rely on their basic model. The interested reader is referred to [4] for more details.

Example 3. This example is a generalization of Example 1. Let $n \leq m$. Let $f : \mathfrak{R}^{m+n} \rightarrow \mathfrak{R}$ and the MPEC be defined by

$$\begin{aligned} & \text{minimize} && f(x, y) = \sum_{i=1}^n (x_i + r_i)^2 + \sum_{j=1}^m (y_j - s_j)^2 - \sum_{i=1}^n r_i^2 - \sum_{j=1}^m s_j^2 \\ & \text{subject to} && 0 \leq (y - (x, 0)) - y \geq 0, \end{aligned}$$

where $r_i = 1$ ($i = 1, \dots, n$), $s_j = 2$ ($j = 1, \dots, m$), and $(x, 0) \in \mathfrak{R}^n \times \mathfrak{R}^{m-n}$. The feasible solution set of this LCP-QP is

$$\{(x, y) : (x_i, y_i) \in S_i, i = 1, \dots, n, y_i = 0, i = n + 1, \dots, m\}$$

where for $i = 1, \dots, n$, $S_i = S_i^1 \cup S_i^2$,

$$S_i^1 = \{(x_i, y_i) : x_i = y_i \geq 0\}, S_i^2 = \{(x_i, y_i) : y_i = 0 \geq x_i\}.$$

The feasible set is the union of 2^n convex polyhedra Ω^k ($1 \leq k \leq 2^n$), which are branches of the feasible set. In each branch, $(x_i, y_i) \in S_i^1$ or $(x_i, y_i) \in S_i^2$ for $i = 1, \dots, n$, and $y_i = 0$ for $i = n + 1, \dots, m$.

Since f is strictly convex, it has a unique global minimizer z^k on each branch Ω^k . In fact, for each k , $z^k = (x^k, y^k)$ where (x_i^k, y_i^k) is either $(-1, 0)$ or $(1/2, 1/2)$ for $i = 1, \dots, n$ and $y_i^* = 0$ for $i = n + 1, \dots, m$. If $j \neq k$, then z^j does not belong to the branch Ω^k , from which it follows that the global minimizer z^j of the objective function f over the branch Ω^j is a local minimizer of the LCP-QP. It is easy to calculate the objective function value at each local minimizer.

The global minimizer is (x^*, y^*) where $x_i^* = -1$ and $y_j^* = 0$ for $i = 1, \dots, n$ and $j = 1, \dots, m$. The global minimum is $-n$.

One may construct more examples which have similar characteristics as this example by choosing different data for r_i and s_j such that $r_i > 0$ and $s_j > 0$.

Example 4. This example is a slight modification of Example 3 by setting $r_i = -1$ ($i = 1, \dots, n$) and $s_j = -2$ ($j = 1, \dots, m$) in the objective function f of Example 3. The feasible set of this LCP-QP is the same as that in Example 3 but the (strictly convex) objective function is such that the unique (local) global minimizer on each branch is the same point, namely the origin. Thus the origin is also a unique (local) global minimizer of this LCP-QP. Furthermore, more examples can be constructed by choosing any negative r_i and s_j .

3. Optimality Conditions

The feasible solution set of the KKT-constrained MP (3) is

$$\mathcal{F} = \{w = (x, y, \lambda) \in Z \times \mathfrak{R}_+^p : (x, y, \lambda) \text{ satisfies (2)}\}. \quad (1)$$

It follows that for each $(x, y, \lambda) \in \mathcal{F}$ we have $(x, y) \in Z$ and $y \in \mathcal{S}(x)$.

Let $w^* = (x^*, y^*, \lambda^*) \in \mathcal{F}$. The tangent cone $\mathcal{T}(w^*, \mathcal{F})$ to \mathcal{F} at w^* is defined by

$$\mathcal{T}(w^*, \mathcal{F}) = \{\lim(w^k - w^*)/\tau^k : w^k \in \mathcal{F} \rightarrow w^*, \tau^k \downarrow 0\}.$$

Write $dw = (dx, dy, d\lambda) \in \mathfrak{R}^n \times \mathfrak{R}^m \times \mathfrak{R}^p$. Then a first-order necessary condition (or stationarity condition) for the feasible point w^* to be a local minimizer of (3) is

$$\nabla f(x^*, y^*)^T (dx, dy) \geq 0, \quad \forall dw \in \mathcal{T}(w^*, \mathcal{F}).$$

As studied in [19], under some MPEC constraint qualifications, the tangent cone $\mathcal{T}(w^*, \mathcal{F})$ coincides with a certain linearized cone of \mathcal{F} at w^* and, therefore, it is possible to obtain a primal-dual formulation of the above stationarity condition which can be used in algorithms for MPEC. QPEC satisfy a natural constraint qualification — their constraints are defined by polyhedral and AVI constraints — so that stationarity is equivalent to primal-dual stationarity; see part (iii) of the next proposition.

To this end, recall some notation introduced in [19]. For any $w^* = (x^*, y^*, \lambda^*) \in \mathcal{F}$, define index sets

$$\begin{aligned} \alpha(w^*) &= \{1 \leq i \leq p : \lambda_i^* = 0 < -(Dx^* + Ey^* + b)_i\} \\ \beta(w^*) &= \{1 \leq i \leq p : \lambda_i^* = -(Dx^* + Ey^* + b)_i = 0\} \\ \gamma(w^*) &= \{1 \leq i \leq p : \lambda_i^* > -(Dx^* + Ey^* + b)_i = 0\} \end{aligned}$$

and the family of index sets

$$\mathcal{A}(w^*) = \{\alpha \subseteq \{1, \dots, p\} : \alpha \supseteq \alpha(w^*), \alpha^c \supseteq \gamma(w^*)\},$$

where α^c is the complement set of α with respect to $\{1, \dots, p\}$. For each $\alpha \in \mathcal{A}(w^*)$, we define a branch of the decomposition as

$$\mathcal{F}_\alpha = \{w \in Z \times \mathbb{R}_+^p : F(x, y) + E^T \lambda = 0, \quad (2)$$

$$\lambda_i = 0 \leq -(Dx + Ey + b)_i, \quad \forall i \in \alpha,$$

$$\lambda_i \geq 0 = (Dx + Ey + b)_i, \quad \forall i \in \alpha^c \}.$$

Using the family of sets $\{\mathcal{F}_\alpha : \alpha \in \mathcal{A}(w)\}$, the feasible set \mathcal{F} of (3) can be locally decomposed at any feasible point and stationarity of (3) can be characterized in terms of traditional nonlinear programming optimality conditions associated associated with each $\{\mathcal{F}_\alpha : \alpha \in \mathcal{A}(w)\}$. These can be stated precisely in the following proposition extracted from [19].

PROPOSITION 1 *Let \mathcal{F} denote the feasible solution set of (3) and $w^* \in \mathcal{F}$.*

(i) *There exists a neighborhood W of w^* such that*

$$\mathcal{F} \cap W = \bigcup_{\alpha \in \mathcal{A}(w^*)} \mathcal{F}_\alpha \cap W. \quad (3)$$

(ii) *The feasible point w^* is a local minimizer of (3) if and only if, for each $\alpha \in \mathcal{A}(w^*)$, w^* is a local minimizer of the nonlinear program*

$$\begin{aligned} & \text{minimize} && f(x, y) \\ & \text{subject to} && w \in \mathcal{F}_\alpha. \end{aligned} \quad (4)$$

(iii) *The feasible point w^* is a stationary point of (3) if and only if, for each $\alpha \in \mathcal{A}(w^*)$, there exist multipliers $\xi \in \mathbb{R}^l$, $\eta \in \mathbb{R}^m$ and $\pi, \zeta \in \mathbb{R}^p$ such that*

$$\begin{aligned} \nabla_x f(x^*, y^*) + G^T \xi - N^T \eta + D^T \pi &= 0, \\ \nabla_y f(x^*, y^*) + H^T \xi - M^T \eta + E^T \pi &= 0, \\ E\eta + \zeta &= 0, \\ \xi \geq 0, (Gx^* + Hy^* + a)^T \xi &= 0, \\ \pi_\alpha \geq 0, (Dx^* + Ey^* + b)_\alpha^T \pi_\alpha &= 0 \\ \zeta_{\alpha^c} \geq 0, \lambda_{\alpha^c}^{*T} \zeta_{\alpha^c} &= 0. \end{aligned} \quad (5)$$

(iv) *The feasible point w^* is a stationary point of (3) if there exist multipliers $\xi \in \mathbb{R}^l$, $\eta \in \mathbb{R}^m$ and $\pi, \zeta \in \mathbb{R}^p$ such that*

$$\begin{aligned} \nabla_x f(x^*, y^*) + G^T \xi - N^T \eta + D^T \pi &= 0, \\ \nabla_y f(x^*, y^*) + H^T \xi - M^T \eta + E^T \pi &= 0, \\ E\eta + \zeta &= 0, \\ \xi \geq 0, (Gx^* + Hy^* + a)^T \xi &= 0, \\ \zeta_i = 0, \quad \forall i \in \gamma(w^*), & \\ \pi_i \geq 0, \zeta_i \geq 0, \quad \forall i \in \beta(w^*), & \\ \pi_i = 0, \quad \forall i \in \alpha(w^*). & \end{aligned} \quad (6)$$

Given $w^* \in \mathcal{F}$, the system (6) is in turn equivalent to the KKT conditions at $w = w^*$ for the following so-called **relaxed nonlinear program**:

$$\begin{aligned}
& \text{minimize} && f(x, y) \\
& \text{subject to} && (x, y) \in Z \\
& && (x, y) \in \mathcal{R}_w = \{(x, y) : F(x, y) + E^T \lambda = 0 \\
& && \quad (Dx + Ey + b)_i = 0, \quad i \in \alpha(w) \\
& && \quad (Dx + Ey + b)_i \leq 0, \lambda_i \geq 0, \quad i \in \beta(w) \\
& && \quad \lambda_i = 0, \quad i \in \gamma(w)\}.
\end{aligned} \tag{7}$$

Part (iv) of the Proposition says that if a feasible point of the MPEC (3) is a KKT point of the relaxed NLP, then it is stationary for the MPEC. Conversely for $w^* \in \mathcal{F}$ and any $\alpha \in \mathcal{A}(w^*)$, it can be proved [19] under linear independence of the active constraint gradients of (4), that solvability of the relaxed KKT system (6) implies solvability of all KKT systems (5), i.e. relaxed stationary points coincide with stationary points.

As it will be seen in the following sections, the relaxed NLP (7) and its KKT conditions (6) play an important role in the development of our generator. In particular for each problem produced by the generator, the associated “solution” vector $w_{\text{gen}} = (x_{\text{gen}}, y_{\text{gen}}, \lambda_{\text{gen}})$ is determined as a feasible point of the QPEC such that $w^* = w_{\text{gen}}$ satisfies (6); hence it is stationary for the QPEC. Since w_{gen} is merely stationary it need not to be a local solution of the QPEC if the objective function is nonconvex. Under convexity of the objective or second-order sufficient conditions [20, 19], w_{gen} will indeed be a local minimizer of the QPEC.

The above analysis of optimality conditions for the AVI-MP also applies to the special cases of Box-MP and LCP-MP. However for the Box-MP and LCP-MP there is no need to introduce the second-level multipliers λ . Consequently, the definitions of index sets $\alpha(z^*)$, $\beta(z^*)$ and $\gamma(z^*)$ (note that z^* replaces w^*) are different in the contexts of the Box-MP and LCP-MP. The details are omitted here.

By definition of the index set $\beta(w^*)$, if $\beta(w^*) = \emptyset$, then the index set $\mathcal{A}(w^*)$ is singleton. In this case, the MPEC (3) is locally a linearly constrained quadratic program. If $\beta(w^*) \neq \emptyset$, the feasible set of the MPEC (3) around w^* is more complicated. Usually any index in $\beta(w^*)$ is called degenerate. Existence of degenerate indices reflects the complexity of MPEC. We formally introduce several definitions of degeneracy.

Definition 1. Suppose $w^* = (x^*, y^*, \lambda^*)$ is a KKT point of the relaxed nonlinear program (7), i.e., w^* is feasible for (7) and there exist multipliers ξ , η , π and ζ such that (6) holds.

- (i) An index i ($1 \leq i \leq l$) is called **first-level degenerate** if $\xi_i = (Gx^* + Hy^* + a)_i = 0$.
- (ii) An index i ($1 \leq i \leq p$) is called **second-level degenerate** if $i \in \beta(w^*)$.
- (iii) A second-level degenerate index i is called **mixed degenerate** if either $\pi_i = 0$ or $\zeta_i = 0$.

By degree of degeneracy of a particular type, we mean the number of degeneracy indices of that type.

We remark that our presentation on optimality conditions, though brief, should be enough for subsequent developments here. Moreover since QPEC are generally much better behaved than (nonlinear) MPEC, we have not discussed MPEC constraint qualifications, or local uniqueness issues for local minimizers, stationary points, or the corresponding Lagrange multipliers. To fully understand optimality conditions of the MPEC, extra care is needed in regard to some of the basic concepts such as tangent cones and certain linearized cones of the MPEC at a feasible point. In particular, these cones are generally nonconvex, for example unions of polyhedral convex cones, in contrast to traditional nonlinear programming where linearized cones are always convex polyhedral. For a detailed treatment of optimality conditions of the MPEC, see [19].

4. QPECgen: A Random Generator of QPEC

The MATLAB code `QPECgen` to be described in this section, and associated documentation, can be obtained from

<http://www.maths.mu.OZ.AU/~danny/qpecgendoc.html>.

4.1. Overview

Once the user has specified the problem type, the problem dimensions, and a number of other problem characteristics mentioned below, `QPECgen` randomly generates the problem matrix data and the “solution” vector $w_{\text{gen}} = (x_{\text{gen}}, y_{\text{gen}}, \lambda_{\text{gen}})$ (for the AVI-constrained case), and then chooses the vector data in such a way that w_{gen} is both feasible for the QPEC and stationary for the corresponding relaxed NLP (7) at $w^* = w_{\text{gen}}$; hence w_{gen} is stationary for the QPEC by Proposition 1(iv). The choice of the vector data must also satisfy degeneracy requirements, after which the various multipliers associated with first- and second-level constraints are determined in order to satisfy (6) with $w = w_{\text{gen}}$. We remark that `QPECgen` does not distinguish between the formulations as an AVI-QP in the form of (1) and the KKT-constrained QP in the form of (3). Strictly speaking, $(x_{\text{gen}}, y_{\text{gen}})$ is the “solution” (stationary point) of the AVI-QP while $w_{\text{gen}} = (x_{\text{gen}}, y_{\text{gen}}, \lambda_{\text{gen}})$ is the “solution” (stationary point) of the KKT-constrained problem (3), where λ_{gen} may not be uniquely determined by $(x_{\text{gen}}, y_{\text{gen}})$. However, we always print w_{gen} in the output file.

We mention several aspects of interest regarding application of algorithms. See Table 1 for more details of the parameters involved.

- Problem types include AVI-QP, Box-QP, LCP-QP, possibly in implicit form; or one of Examples 3 or 4. Parameters: `qpec.type`, `implicit`.
- Degeneracy, particularly in the second-level, and the degree of degeneracy of the MPEC at a solution point can have a significant impact on the behavior

of certain algorithms such as PSQP, see Section 5. Parameters: `first_deg`, `second_deg`, `mixed_deg`, `tol_deg`.

- Ill-conditioning of the Hessian of the objective function is likely to create difficulties such as slow convergence behaviour for most MPEC algorithms. Similarly, ill-conditioning of the matrix M in the second-level function F also affects the nature of the MPEC in terms of stability of its constraints, hence the performance of some algorithms. Different scalings of the objective and constraint data also affects performance. Well-scaled problems can be generated by choosing these parameters of moderate size and similar magnitude. Parameters: `cond_P`, `cond_M`, `scale_P`, `scale_M`.
- Convexity of the objective function is also considered. Though this is useful for instance for methods like PSQP that tend to find stationary points (which then must be local minimizers), the MPEC may still have local minima which are not global minima. Parameter: `convex_f`.
- Symmetry and monotonicity properties of the matrix M in the second-level function F are very useful but not always present. Certain monotonicity properties are used to prove convergence of some algorithms for solving complementarity problems [6] and variational inequalities, and monotonicity is also very helpful in finding feasible points when the first-level constraints reduce to constraints on the first-level variables only, i.e. $Z = X \times \mathbb{R}^m$ for some $X \subset \mathbb{R}^n$. Therefore, positive semidefiniteness (monotonicity) of M is likely to improve the performance of the MPEC algorithms. Symmetry and positive semidefiniteness of M mean that the QPEC is equivalent to a bilevel programming problem. Parameters: `symm_M`, `mono_M`.

The random seed in MATLAB can be redefined by specifying the option `rand_seed` if the user wishes. By choosing different random seed numbers, the user can generate different QPEC examples without changing other problem parameters. Also, the output format is determined by the parameter `output`.

4.2. Input Data File and Data Consistency

The user needs to initialize, in MATLAB, the parameters used by the generator. We generally do this by writing or modifying a MATLAB M-file called `parameter` (we omit the `.m` suffix) which contains data necessary for the main program `QPECgen` to run. The variables used in `parameter` are listed in Table 1.

The parameter settings must satisfy certain rules are explained in Table 1; let us see some examples. The parameter `cond_P` represents the condition number of the matrix P and should be a real number not less than 1. The parameters `convex_f`, `symm_M`, `mono_M` and `implicit` are binary variables with values 1 or 0 only. The parameters `first_deg` and `second_deg` should not exceed the number of first-level and second-level constraints respectively. The parameter `mix_deg` denotes the cardinality of the mixed degenerate index set and should not be greater than `second_deg` since every mixed degenerate index should be a second-level degenerate

Table 1. Input Data

qpec_type	Indicate special types of QPEC problems. qpec_type = 100, 200, 300, 800, 900 means the AVI-QP, Box-QP, LCP-QP, Examples 3 and 4 respectively.
n	The dimension of the first-level variable x .
m	The dimension of the second-level variable y .
l	The number of the first-level inequality constraints.
p	The number of the second-level inequality constraints for the AVI-QP.
cond_P	The condition number of the matrix P .
scale_P	Approximate magnitude of maximum singular value of P .
convex_f	This is a binary element. convex_f = 1 means that f is convex, and convex_f = 0 means it is not necessarily convex.
symm_M	This is a binary element. If symm_M = 1, the matrix M is symmetric. If symm_M = 0, the matrix M is asymmetric.
mono_M	This is a binary element. mono_M = 1 indicates that M is monotone, and mono_M = 0 not necessarily monotone.
cond_M	The condition number of the matrix M .
scale_M	Approximate magnitude of maximum singular value of M .
second_deg	The cardinality of the second-level degenerate index set.
first_deg	The cardinality of the first-level degenerate index set.
mix_deg	The cardinality of the mixed degenerate index set.
tol_deg	A small positive scalar used to determine approximate degeneracy, e.g. 10^{-6} (see §5.2.1).
implicit	A binary variable. implicit = 1 or 0 denotes that the first-level constraints involve the variable x only, or both variables x and y , respectively.
rand_seed	A nonnegative integer setting the seed for the random no. generator.
output	The number to control data output formats. output = 1 for MAT-file called QPECgen_data , 2 for ASCII-file called QPECgen_ascii , and 3 for both.

index. If the data supplied by the user are inconsistent, QPECgen will print a warning message like

Warning: Wrong data for qpec_type.

In this case the generator will attempt to choose a correct default value for this parameter, allowing the generator to run successfully.

4.3. The Generator QPECgen

We now present an outline of the method for generating quadratic programs with affine variational inequality constraints (1) or (3), as implemented in MATLAB.

Algorithm: QPECgen

Step 1. Retrieve input parameters from `parameter`.

Step 2. Check consistency of input parameters; redefine inconsistent values if necessary.

Step 3. Set the random seed in MATLAB environment as specified in `parameter`; zero is the default value.

Step 4. Generate the symmetric matrix P in the objective function f randomly as $P = \mathbf{rand}(n+m) - \mathbf{rand}(n+m)$ and $P = P + P'$. If `convex_f` = 1, do Schur decomposition [15] for P and form a new positive definite matrix P by shifting the eigenvalues such that its minimum eigenvalue is uniformly distributed in $[0, 1]$. Next a new matrix is formed with the required condition number `cond_P` by retaining its minimum eigenvalue and redistributing other eigenvalues. Finally, scale the matrix by $\frac{\mathbf{scale_P}}{\mathbf{cond_P}}$ so that the magnitude of maximum singular value is roughly equal to `scale_P`. If `convex_f` = 0, the required matrix can be constructed by doing singular value decomposition [15] and using the similar techniques above. **Exit:** P .

Step 5. If `implicit` = 2, generate the matrix $A = [G; H]$ of the first-level constraints by $A = \mathbf{rand}(l, m+n) - \mathbf{rand}(l, m+n)$. If `implicit` = 1, generate A by $A = [\mathbf{rand}(l, n) \text{ zeros}(l, m)]$. **Exit:** A .

Step 6. Generate the matrices M and N in the second-level objective function. For the matrix M , the properties of symmetry and monotonicity and also its condition number and scaling of M such that its magnitude of maximum singular value is roughly equal to `scale_M` can be determined similar to Step 4, though a little more care has to be taken in the asymmetric case. **Exit:** M, N .

Step 7. Generate an “optimal solution” randomly by $(x_{\text{gen}}, y_{\text{gen}}) = \mathbf{rand}(n+m, 1) - \mathbf{rand}(n+m, 1)$. In the case of the Box-QP and LCP-QP, y_{gen} should be feasible to the second-level constraints. For example, y_{gen} is nonnegative for the LCP-QP. **Exit:** $(x_{\text{gen}}, y_{\text{gen}})$.

Step 8. Randomly generate the matrices D and E used for the constrained region of the (second-level) AVI. Let the indices $1, \dots, \mathbf{second_deg}$ be degenerate, the next $p_{\text{nonactive}}$ indices be nonactive, where $p_{\text{nonactive}}$ is a random nonnegative integer not greater than $p - \mathbf{second_deg}$, and the remaining indices be active but nondegenerate for the second-level constraints (i.e., their corresponding second-level multipliers are strictly positive). Generate the second-level multipliers λ_{gen} and the vector b in the second-level constraints satisfying the KKT

conditions of the second-level problem AVI at $(x_{\text{gen}}, y_{\text{gen}})$ with the required degeneracy conditions just specified. By the KKT conditions of the second-level problem, determine the vector q in the second-level function. **Exit:** $D, E, b, q, \lambda_{\text{gen}}$.

Step 9. Consider the first-level constraints. The indices $1, \dots, \text{first_deg}$ correspond to degenerate constraints, the next l_nonactive constraints are non-active, where l_nonactive is a nonnegative integer with $\text{l_nonactive} \leq l - \text{first_deg}$, and the remaining first-level constraints are active but nondegenerate. Generate the first-level multipliers ξ associated with the first-level constraints and the vector a in the first-level constraints satisfying the above degeneracy conditions and the KKT conditions of the relaxed NLP. **Exit:** a .

Step 10. Calculate the index sets $\alpha(w^*), \beta(w^*)$ and $\gamma(w^*)$ with $w^* = (x_{\text{gen}}, y_{\text{gen}}, \lambda_{\text{gen}})$. At the same time, let the first mix_deg indices in the set $\beta(w^*)$ be degenerate in the KKT conditions of the relaxed NLP. Generate η, π and ζ satisfying the above degeneracy conditions and the KKT conditions of the relaxed NLP.

Step 11. Determine the vectors c and d in the objective function f by the KKT conditions of the relaxed NLP. **Exit:** c, d .

Step 12. Store the generated data: $P, c, d, A, a, N, M, q, D, E, b, x_{\text{gen}}, y_{\text{gen}}, \lambda_{\text{gen}}$.

Remarks.

- (i) The methods for generating Box-QP and LCP-QP are similar to the above.
- (ii) The data generated by `QPECgen` is stored in a MAT-file (internal MATLAB format) called `QPECgen_data` and/or an ASCII file `QPECgen_ascii`. Advantages of ASCII output are that the user can view or edit the generated data outside MATLAB and, more importantly, these data can be easily used to test MPEC codes written in other languages rather than MATLAB.
- (iii) As already mentioned, the generated solution vector w_{gen} , which is included in the output data file, is a stationary point of the QPEC but may not be a local minimizer. Thus the corresponding objective function value can only be used as a guide to the quality of the solution of an MPEC algorithm; indeed there may be other stationary points and multiple local minimizers as shown in the examples of Section 2.
- (iv) Examples 3 and 4 in Section 2 are generated by `QPECgen` by choosing `qpec_type` = 900 and 800 respectively. We remark for these two examples that only the two parameters n and m play a role in `QPECgen` because all other parameters are fixed in accordance with these examples.

5. Testing Algorithms on Medium-Small Problems

This section is devoted to testing some existing MPEC algorithms on problems produced by the generator `QPECgen`. We first describe the algorithms to be tested, confining discussion to MPEC with linear constraints, that is polyhedral convex

first-level constraints and affine VI equilibrium or LCP-constraints, aimed of course at QPEC, see (3). (However, each of these algorithms can be extended to MPEC with nonlinear first- and second-level constraints.) Then the implementation details are addressed. Lastly, numerical results are reported.

Our largest problem in the AVI-QP format has $n + m = 70$ variables (x, y) and $l + m + 2p = 106$ constraints excluding the complementarity conditions $\lambda_i g_i(x, y) = 0$. Our largest problem in the LCP-QP format has $n + m = 208$ variables (x, y) and $l + 2m = 404$ constraints excluding the complementarity conditions $y_i F_i(x, y) = 0$.

We stress that the purpose of this section is to stimulate further algorithmic and computational improvements in the field, rather than to provide definitive data on algorithm behaviour. Specifically, our problems are randomly generated QPEC of small to medium size, our implementations and application of codes are preliminary, and the parameter settings for these methods are subject to further testing. For instance, a possible advantage of PIPA that is not seen in the results below but that we have observed in preliminary computation on an electricity grid application [17] is its tendency to avoid local solutions more readily than PSQP. Collation of results on this and other real-world problems is work in progress. Large sparse QPEC and MPEC with nonlinear objective or constraint functions also require investigation.

One word for notation used in the sequel. We reserve α , β and γ for index sets when the decomposition is carried out, and λ for the Lagrange multiplier of the second-level problem in the AVI-MP (3).

5.1. Description of Algorithms to be Tested

5.1.1. Nonlinear Programming Methods As discussed in Section 2, the AVI-MP can be reformulated as the nonlinear programming problem (3) if the second-level problem of the AVI-MP is written in the KKT form. So it is quite natural to apply standard and well-developed nonlinear programming approaches to (3). Unfortunately such approaches often prove to be somewhat naive and even theoretically and numerically inappropriate. The reason is that the usual constraint qualifications are not satisfied for (3) even under some very strong assumptions [5, 29], for example, the objective function is strictly convex in (x, y) and the second-level problem AVI is also strongly monotone with respect to y . Some examples are presented in [9, 19]. Therefore, standard nonlinear programming approaches are not necessarily reliable; see the problem below for example. Further work will no doubt improve this situation. In any case, we include the naive nonlinear programming approach for comparison.

Consider the following LCP-QP problem

$$\begin{aligned} \text{minimize} \quad & 0.5(x^2 + y^2) + x - y \\ \text{subject to} \quad & 0 \leq y - (-x + y) \leq 0. \end{aligned}$$

It is easy to see that the above problem has only one local (global) solution $(-1, 0)$. We have tested this example using a MATLAB nonlinear programming solver `constr`, and MINOS linked to MATLAB. Both codes converge to the nonstationary

point $(0,0)$ when the starting point is chosen to be in \mathbb{R}_+^2 and close to $(0,0)$, say within the Euclidean ball of radius 10^{-4} .

5.1.2. A Piecewise Sequential Quadratic Programming (PSQP) Method Sequential Quadratic Programming, SQP, and its variants are amongst the most important and most popular methods for general nonlinear programs. This is a consequence of their fast local convergence, amenability to application of quasi-Newton techniques, and globally convergent extensions; see [11].

Piecewise sequential quadratic programming, PSQP, [19, 20, 28], is an extension of SQP methods to MPEC via disjunctive decomposition; for example, Proposition 1 (ii) shows that solving (3) is equivalent to solving finitely many linearly constrained nonlinear programs. See [19] for application of PSQP to nonlinearly constrained MPEC.

It seems natural when solving AVI-MP (3) to require feasibility of each iteration point, similar to solving linearly constrained NLP. A “phase I” method attempts to find a feasible solution of (3); “phase II” then attempts to solve the AVI-MP by generating a sequence of feasible iterates whose corresponding objective function values decrease monotonically to a stationary point or local minimizer. The phase I method we use attempts to solve the following quadratic program:

$$\begin{aligned} & \underset{(z,\lambda)}{\text{minimize}} && -\lambda^T g(z) \\ & \text{subject to} && z \in Z \\ & && F(z) + E^T \lambda = 0 \\ & && g(z) \leq 0, \quad \lambda \geq 0. \end{aligned} \tag{1}$$

If the objective function value of (1) corresponding to a solution (z, λ) is zero, then this point is global minimum of (1) and is feasible for (3). See also [13] for a sufficient condition for success of the phase I method in the sense that every stationary point of (1) is a global minimizer with optimal value 0.

The algorithm proposed in [19, 28] is restated below. Each iteration depends on the current, feasible iterate $w^k = (z^k, \lambda^k)$, and the solution of a quadratic programming subproblem

$$\begin{aligned} & \underset{(d,\lambda)}{\text{minimize}} && \nabla f(z^k)^T d + \frac{1}{2} d^T \nabla^2 f(z^k) d \\ & \text{subject to} && (z^k + d, \lambda) \in \mathcal{F}_{\alpha^k} \end{aligned} \tag{2}$$

where \mathcal{F}_{α^k} is the set defined in (2) with $\alpha = \alpha^k$ and $w^* = w^k$, and α^k is a member of $\mathcal{A}(w^k)$. Since w^k is feasible, it follows that \mathcal{F}_{α^k} is nonempty, i.e. (2) is feasible. Furthermore if f is quadratic as in the QPECgen problems tested later, then we can use $f(z^k + d)$ instead of the objective of (2).

By the *relaxed quadratic program* at w^k we mean the problem derived from (2) by replacing the feasible region \mathcal{F}_{α^k} with the relaxed feasible region \mathcal{R}_{w^k} defined as in (7) with $w^* = w^k$:

$$\begin{aligned} & \underset{(d,\lambda)}{\text{minimize}} && \nabla f(z^k)^T d + \frac{1}{2} d^T \nabla^2 f(z^k) d \\ & \text{subject to} && (z^k + d, \lambda) \in \mathcal{R}_{w^k} \end{aligned} \tag{3}$$

The idea of approximate stationarity is used in the stopping criteria of the method. We say that $(z, \lambda_h, \lambda_g)$ is an ε -KKT point of the nonlinear programming problem

$$\begin{aligned} & \text{minimize} && f(z) \\ & \text{subject to} && h(z) = 0 \\ & && g(z) \geq 0, \end{aligned}$$

for suitable smooth functions f , g and h if

$$\|\nabla f(z) + \nabla h(z)^T \lambda_h + \nabla g(z)^T \lambda_g\| + \|h(z)\| + \|\min(-g(z), \lambda_g)\| \leq \varepsilon.$$

It can be easily seen that an ε -KKT point with $\varepsilon = 0$ is a KKT point of this NLP and vice versa. We also say that z is ε -stationary for this NLP if there exist λ_h and λ_g such that $(z, \lambda_h, \lambda_g)$ is an ε -KKT point; likewise, a feasible point w^* of (3) is ε -stationary for (3) if it is ε -stationary for each NLP (4) where $\alpha \in \mathcal{A}(w^*)$.

Note for Step 1 below that $(0, \lambda^k, \xi, \eta, \pi, \zeta)$ is an ε -KKT point of (2) if and only if $(w^k, \xi, \eta, \pi, \zeta)$ is an ε -KKT point of (4) with $\alpha = \alpha^k$.

Algorithm: PSQP

Step 0. Let $t \in (0, 1)$, $\kappa \in (0, 1/2)$ and $\varepsilon \geq 0$ be constants.

Find a feasible point $w^0 = (z^0, \lambda^0) \in \mathfrak{R}^{n+m} \times \mathfrak{R}^p$ of (3) by solving (1).

Let $k = 0$ and $\mathcal{A}^0 = \mathcal{A}(w^0)$.

Step 1. (*Direction finding.*) Choose an index set $\alpha^k \in \mathcal{A}^k$ and let $\mathcal{A}^k = \mathcal{A}^k \setminus \{\alpha^k\}$. If the quadratic program (2) is unbounded below, then STOP; otherwise find a solution (d^k, λ^+) of (2), with multipliers ξ, η, π, ζ . If $(0, \lambda^k, \xi, \eta, \pi, \zeta)$ is an ε -KKT point of (2) then go to Step 3.

Step 2. (*Serious step.*) Do a line search, i.e., let τ_k be the largest member of $\{1, t, t^2, t^3, \dots\}$ such that

$$f(z^k + \tau_k d^k) - f(z^k) < \kappa \tau_k \nabla f(z^k)^T d^k.$$

Let $w^{k+1} = (x^{k+1}, y^{k+1}, \lambda^{k+1}) = w^k + \tau_k (d^k, \lambda^+ - \lambda^k)$, and $\mathcal{A}^{k+1} = \mathcal{A}(w^{k+1})$.

Let $k = k + 1$ and go to Step 1.

Step 3. (*Stopping rule.*) If the stopping condition is satisfied then STOP.

Step 4. (*Null step.*) Let $w^{k+1} = w^k$, $\mathcal{A}^{k+1} = \mathcal{A}^k$, $k = k + 1$, and go to Step 1.

The line search in Step 2 is an attempt to globalize the original version of PSQP [28, 19], where local convergence of the algorithm is established. It can be justified theoretically when $\nabla^2 f(z^k)$ is positive definite or is replaced in (2) by a positive definite matrix. The line search is not strictly necessary when f is quadratic and possibly nonconvex; see the remark following Proposition 2 to follow. However it can help stabilize PSQP if the QP solver is having numerical difficulties, such as occasionally occur for the QP solver in the MATLAB 4.2 Optimization Toolbox.

We present three straightforward stopping rules for Step 3. Each guarantees that the algorithm will only terminate if the current point z^k is approximately stationary for each NLP branch (4) at z^k , or if the QPEC is unbounded below.

Stopping rule A terminates the algorithm if all branches of the feasible region at the current point w^k have been checked and w^k is ε -stationary for each of them.

Stopping condition A. $\mathcal{A}^k = \emptyset$.

Stopping rule B, from [20], is slightly more sophisticated. It uses the knowledge from Proposition 1 that since $w^k \in \mathcal{F}$, then w^k is an (approximate) stationary point of the MPEC if it is an (approximate) stationary point of (7). This last condition is also equivalent to (d, λ^k) with $d = 0$ being an (approximate) stationary point of (3).

Stopping condition B. Either $\mathcal{A}^k = \emptyset$ or $(0, \lambda^k, \xi, \eta, \pi, \zeta)$ is an ε -KKT point of (3).

Stopping rule B is a heuristic to improve on the exhaustive approach of stopping rule A. While it is not guaranteed to reduce the number of QP solves per serious step, it costs almost no more to use than stopping rule A and appears to be quite effective [20]; see later results also.

For completeness we mention a third alternative that will not be implemented because it takes too much advantage of QPECgen, as explained below. Note that w^k is a stationary point of (7) if and only if $(0, \lambda^k)$ is a stationary point of (3). These equivalent statements can be checked by solving a linear programming feasibility problem, namely checking whether there exist multipliers (ξ, η, π, ζ) satisfying (6) when $w^* = w^k$. An ε -approximate version based on linear programming can also be developed.

Stopping condition C. $\mathcal{A}^k = \emptyset$, or $(0, \lambda^k)$ is a stationary point of (3).

Unlike stopping conditions A and B, stopping condition C is guaranteed to hold for any feasible stationary point w^k of the MPEC that is stationary for the relaxed NLP. Hence stopping rule C is guaranteed to immediately identify the generated “solution” vector w_{gen} of any QPECgen problem if $w^k = w_{\text{gen}}$. So we have not implemented it because this would prevent us from observing the effect of second-level degeneracy at solution points on PSQP for all QPECgen problems. Instead we compare the effect of second-level degeneracy via stopping rules A and B.

PROPOSITION 2 *Consider an MPEC in the form (3) where the objective function f is strictly convex and quadratic. Assuming w^0 is feasible for this problem, then PSQP terminates after finitely many steps at an ε -stationary point. If $\varepsilon = 0$, termination occurs at a local minimizer of the QPEC.*

Proof: Since f is strictly convex and quadratic, for each k we have $w^k \in \mathcal{F}_{\alpha^k}$, $\tau_k = 1$ and $w^{k+1} \in \mathcal{F}_{\alpha^k}$. Furthermore w^{k+1} is a global minimizer of f on the branch \mathcal{F}_{α^k} . The result is then clear because we cannot visit any branch (2) of the

feasible region more than once, since $f(w^{k+1}) < f(w^k)$, and there are only finitely many branches. ■

Remark: In fact, the above PSQP method can be adapted to solving any QPEC, even if f is nonconvex, as follows. First, omit the line search (i.e., by taking $\tau_k = 1$ at each iteration). Second, assume in Step 1 that if the QP solver returns $(d^k, \lambda^+) \neq (0, \lambda^k)$, then (d^k, λ^+) is a stationary point of (2) with a strictly lower objective value than at $(0, \lambda^k)$. Then PSQP terminates finitely if each QP (2) has only finitely many stationary points. (Actually we believe finite termination occurs for *any* AVI-QP. The proof relies on a claim we will not prove here to conserve space, that the objective function of a quadratic program takes only finitely many values on the set of all KKT points of the QP, even if there are infinitely many KKT points.)

We note an alternative phase I for implicit QPEC, i.e. when H is a zero matrix so that the first-level constraints reduce to $Gx + a \leq 0$. First select x satisfying the first-level constraints, an easy application of linear programming for instance, and then attempt to solve the lower-level AVI problem in y . In the special case of implicit programs with LCP constraints, a standard algorithm such as Lemke's method [6] can be applied to (attempt to) solve the lower-level problem. See Subsubsection 5.2.3.

5.1.3. A Penalty Interior-Point Algorithm (PIPA) Interior-point methods have become extremely popular for linear programming since the 1980's, and subsequently also proved to be powerful for linear and nonlinear complementarity, and related variational inequalities too. As the second-level problem of MPEC is a variational inequality problem or one of its special cases, interior-point ideas can be beneficial in MPEC methods.

PIPA is the most extensively treated algorithm proposed in the monograph [19]; it is an infeasible interior-point method for the implicit MPEC with mixed nonlinear complementarity constraints. See Subsubsection 5.2.3 for an implementation of this method for implicit LCP-QP which we call i-PIPA.

Here we describe a feasible-point variant that allows for first-level constraints jointly on (x, y) .

Assume $(x, y, \lambda, v) \in \mathfrak{R}^{n+m+2p}$ is a strictly feasible point in the sense that

$$\begin{aligned} Gx + Hy + a &\leq 0 \\ Nx + My + q + E^T \lambda &= 0 \\ Dx + Ey + b + v &= 0 \end{aligned}$$

and

$$\lambda \circ v > \rho \mu e,$$

where $\mu = \lambda^T v / p$, ρ is a positive constant, $e \in \mathfrak{R}^p$ is the vector consisting of ones, and $a \circ b = (a_1 b_1, \dots, a_p b_p)$ is the Hadamard product of vectors $a, b \in \mathfrak{R}^p$.

The central part of PIPA is to solve the following quadratic program:

$$\begin{aligned}
& \underset{d \in \mathfrak{R}^r}{\text{minimize}} && \nabla f(x, y)^T(dx, dy) + \frac{1}{2}d^T Qd \\
& \text{subject to} && Gdx + Hdy \leq -a - Gx - Hy \\
& && Ndx + Mdy + E^T d\lambda = 0 \\
& && Ddx + Edy + dv = 0 \\
& && \text{diag}(v)d\lambda + \text{diag}(\lambda)dv = -\lambda \circ v + \sigma\mu e,
\end{aligned} \tag{4}$$

where $d = (dx, dy, d\lambda, dv) \in \mathfrak{R}^r$, $Q \in \mathfrak{R}^{r \times r}$ with $r = n + m + 2p$; \mathfrak{R}_{++}^p denotes the positive orthant of \mathfrak{R}^p ; and $\text{diag}(v)$ is a diagonal matrix with the diagonal elements v_1, \dots, v_p .

Under suitable conditions, the quadratic program (4) has a (unique) solution. See [19] for the case of implicit programs and [13] for the general case. Let d be a solution of (4). Following the spirit of some interior-point methods, the iteration sequence has to be in a neighbourhood of the so-called central path along with other properties. To this end, each step size is strictly bounded above by the largest $\tau \in (0, 1]$ such that

$$\begin{aligned}
& \lambda + \tau d\lambda \geq 0 \\
& v + \tau dv \geq 0 \\
& (\lambda + \tau d\lambda) \circ (v + \tau dv) \geq \rho\mu e \\
& \mu(\tau) = (\lambda + \tau d\lambda)^T (v + \tau dv) / p \geq (1 - \tau)\mu
\end{aligned} \tag{5}$$

where $\rho > 0$ is a constant associated with the neighborhood of the central path. Taken strictly, the roles of each inequality in (5) are explained as follows. The first two inequalities ensure that the new iterate is still in the interior of \mathfrak{R}_{++}^{2p} . The third inequality implies that each component $(\lambda_i + \tau d\lambda_i)(v_i + \tau dv_i)$ of the vector at left has the same order of magnitude as μ . The fourth one guarantees that the complementarity gap goes to zero as the algorithm progresses.

The step size, in addition to satisfying the interiority and centrality conditions (5), must be chosen with the objective function in mind, hence uses the penalized objective function of the MPEC which is defined as follows: for $w = (x, y, \lambda, v) \in \mathfrak{R}_{++}^r$,

$$P_\theta(w) = f(x, y) + \theta\phi(w), \tag{6}$$

where $\theta > 0$ is a penalty parameter, and

$$\phi(w) = v^T \lambda.$$

We are now ready to state a penalty interior-point algorithm that has a simpler form than the algorithm of [19], though in contrast to [19] we offer no convergence proof at this point. However, the feasibility issue of the following algorithm has been studied in [13], and its global convergence is under investigation.

Algorithm: PIPA

Step 0. Let $\sigma_0 \in [0, 1)$, $\rho \in (0, 1)$, $\kappa \in (0, 1)$, $t \in (0, 1)$, $\bar{\theta} > 1$ and $\theta_0 > 1$.

Let $w^0 = (x^0, y^0, \lambda^0, v^0) \in \mathfrak{R}^r$ be a strictly feasible starting point in the sense described above. Let $\varepsilon_1 > 0$ and $\varepsilon_2 > 0$ be two small constants.

Let $Q_0 \in \mathfrak{R}^{r \times r}$ be a positive definite matrix. Let $k = 0$.

- Step 1.** (*Direction finding.*) Solve (4) with $w = w^k = (x^k, y^k, \lambda^k, v^k)$, $Q = Q_k$, $\mu = \mu_k$ and $\sigma = \sigma_k$. Let $d^k = (dx^k, dy^k, d\lambda^k, dv^k)$ be the unique solution of (4).
- Step 2.** (*Stopping rule.*) If $0 \in \mathfrak{R}^r$ is an ε_1 -stationary point of (4), and $\mu_k < \varepsilon_2$ then STOP.
- Step 3.** (*Penalty update.*) Let θ_{k+1} be the largest number in $\{\theta_k, \theta_k \bar{\theta}, \theta_k \bar{\theta}^2, \dots\}$ such that
- $$\nabla f(x^k, y^k)^T(dx^k, dy^k) + \theta_{k+1} \nabla \phi(w^k)^T d^k < -\phi(w^k).$$
- Step 4.** (*Line search.*) Let τ be the largest number in $\{t, t^2, t^3, \dots\}$ strictly satisfying (5). Let χ_k be the largest number in $\{\tau, \tau t, \tau t^2, \dots\}$ satisfying
- $$P_{\theta_{k+1}}(w^k + \chi_k d^k) - P_{\theta_{k+1}}(w^k) \leq \kappa \chi_k [\nabla f(x^k, y^k)^T(dx^k, dy^k) + \theta_{k+1} \nabla \phi(w^k)^T d^k].$$
- Step 5.** Let $w^{k+1} = w^k + \chi_k d^k$. Let $\mu_{k+1} = (\lambda^{k+1})^T v^{k+1} / p$, and $Q_{k+1} \in \mathfrak{R}^{r \times r}$ be a positive definite matrix. Choose $\sigma_{k+1} \in [0, 1)$. Let $k = k + 1$, and go to Step 1.

Remarks:

(i) The PIPA above starts with a strictly feasible starting point which can be found by solving a quadratic program similar to (1) but with some modifications. For example, include small strictly positive lower bounds as constraints on each component of y and λ . More precisely, one may obtain such a strictly feasible starting point by solving the following quadratic program

$$\begin{aligned} & \underset{(z, \lambda)}{\text{minimize}} && -\lambda^T g(z) + 10^{-3}(z, \lambda)^T(z, \lambda) \\ & \text{subject to} && z \in Z \\ & && F(z) + E^T \lambda = 0 \\ & && g(z) \leq -10^{-3}e, \quad \lambda \geq 10^{-3}e. \end{aligned} \tag{7}$$

If the second-level variable y is not involved in the first-level constraints, QPEC becomes an implicit program. Similar to PSQP, we may use Lemke's method to find a strictly feasible starting point in the case of implicit programs LCP-MP. See Subsubsection 5.2.3.

(iii) Global convergence of PIPA has been established in [19] under suitable assumptions including a kind of generalised P-matrix property for the second-level problem and the requirement that $Z = X \times \mathfrak{R}^m$ for some polyhedral set $X \subset \mathfrak{R}^n$.

(iv) If the QPEC is an implicit program and M is monotone, then dv and $d\lambda$ in the quadratic program (4) can be uniquely determined as linear functions of dx and dy . In this case, suitable implementations of the quadratic program (4) can save computational costs, see Subsubsection 5.2.3 for implementation details.

5.2. Some Details of Implementation and Numerical Results

In this subsection, we describe some more details about the methods we are going to test. All computation is carried out on a SPARC 10 by using MATLAB version

4.2c. This subsection involves some standard MATLAB terminology which will not be explained here; see [22] for more details.

5.2.1. Implementation of QPEC with joint upper-level constraints: AVI-QP

The Nonlinear Programming Approach

We use two nonlinear programming codes: `constr` from the MATLAB optimization toolbox, and a version of MINOS which is connected via the MEX interface to MATLAB. By NLP/`constr` we mean the application of `constr` to the KKT-constrained MP (3). NLP/MINOS means the application of MINOS to (3).

The M-file `constr` solves constrained nonlinear programs based on the sequential quadratic programming method. To support `constr`, the user is required to create a main M-file (say `nlpavi`) which initializes the testing example and invokes `constr`, an M-file (`fun_nlpavi`) which defines the objective function and the constraints, and an M-file (`grad_nlpavi`) which defines gradients of the objective function and constraints if the user wishes to do so. The default parameter values in `constr` are used.

MINOS is a widely used nonlinear programming code [24], which we apply to test the naive nonlinear programming method. Unlike `constr`, MINOS uses a projected augmented Lagrangean method in which a linearly constrained nonlinear program is formed at each iteration and then solved by a reduced-gradient algorithm in conjunction with quasi-Newton techniques. MINOS is originally written in Fortran. We use a version linked to MATLAB.

To support MINOS, we need to create a main M-file (say `minosavi`) which initializes the test example and invokes MINOS, an M-file (`obj_minosavi`) which defines the objective function and its gradient, and an M-file (`con_minosavi`) which defines the constraints and the corresponding Jacobian mapping. One may observe that `constr` and MINOS access the functions and their derivatives in different ways: `constr` requires the objective function and constraint functions to reside in one file, and all their derivatives to reside in another file; while MINOS extracts the objective function and its derivative from one file, and has the constraints and their derivatives in another file.

Piecewise Sequential Quadratic Programming Method

The parameters used are $\kappa = 0.1$ and $t = 0.5$ in the line search, step 2, and $\varepsilon = 10^{-6}$ in the stopping criteria. The degeneracy tolerance is `tol_deg` = $\delta = 10^{-6}$. Here we offer an explanation of approximate degeneracy or approximate decomposition. Given a small positive tolerance `tol_deg` = δ and a feasible point $w^* = (x^*, y^*, \lambda^*)$, define approximate index sets

$$\begin{aligned}\alpha(w^*, \delta) &= \{1 \leq i \leq p : -\lambda_i^* - (Dx^* + Ey^* + b)_i > \delta\} \\ \beta(w^*, \delta) &= \{1 \leq i \leq p : |\lambda_i^* + (Dx^* + Ey^* + b)_i| \leq \delta\} \\ \gamma(w^*, \delta) &= \{1 \leq i \leq p : \lambda_i^* + (Dx^* + Ey^* + b)_i > \delta\}\end{aligned}$$

and the family of index sets

$$\mathcal{A}(w^*, \delta) = \{\alpha \subseteq \{1, \dots, p\} : \alpha \supseteq \alpha(w^*, \delta), \alpha^c \supseteq \gamma(w^*, \delta)\},$$

where α^c is the complement set of α with respect to $\{1, \dots, p\}$. When $\delta = 0$, these approximate index sets reduce to the exact index sets defined in Section 3. The family of index sets $\mathcal{A}(w)$ is replaced by the approximate family of index sets $\mathcal{A}(w, \delta)$ when choosing which subproblem (2) to solve in PSQP.

A feasible starting point is generated by solving (1) with any starting point supplied by the user.

The quadratic programming routine used is the M-file `qp` from the MATLAB optimization toolbox. Note that `qp` does not have a maximum inner iteration number. We impose a maximum inner iteration number as 2000, which means that the solution provided by `qp` is unreliable if the routine terminates after 2000 inner iterations.

The main M-file to implement PSQP for the AVI-QP is `psqpavi`, which requires three M-files, `f`, `df` and `d2f` for evaluating the objective function, the gradient of objective function and the Hessian of the objective function, respectively. This breakdown of M-files for the objective function and its derivatives is overkill in the case of QPEC since f is then quadratic; however the code is designed for more general problems when f is nonlinear.

Two different versions of PSQP are implemented. The only difference of these two versions is that we use different stopping conditions, i.e., **Stopping condition A** and **Stopping condition B**.

Penalty Interior-Point Algorithm

The parameters used are $\kappa = 0.1$, $\varepsilon_1 = 10^{-6}$, $\varepsilon_2 = 10^{-8}$, $t = 0.95$, $\rho = 0.02$, $\sigma_k \equiv 0.3$, $\bar{\theta} = 1.2$, $\theta_0 = 1.2$, Q_k is chosen as follows,

$$Q_k = \begin{pmatrix} \nabla^2 f(z^k) & 0 \\ 0 & \mu_k I \end{pmatrix}$$

where I is the identity matrix of dimension $2p$.

A strictly feasible starting point is generated by solving (7) from any starting point supplied by the user. This QP and the QP in Step 1 of the method are solved by the MATLAB solver `qp`.

The main M-file, which we call `pipaavi`, uses the M-files `f`, `df`, `d2f` and `pipapenavi`, the first three of which play the same roles as for PSQP. The last file provides the penalty function P_θ .

5.2.2. Numerical results for QPEC with joint upper-level constraints: AVI-QP
We present numerical results on four test sets each consisting of four randomly generated AVI-QP problems, a total of sixteen problems. For each set of four problems, the same parameter settings are passed to `QPECgen`.

Set 1, Problems 1–4. The parameters for our first test set are set up as in Table 2. For this set of parameters, the QPEC has very nice properties in the sense that the objective function is strictly convex with respect to both variables x and y , the Hessian of the objective function is well-conditioned, the second-level problem is symmetric, strongly monotone and well-conditioned with respect to the second-level

Table 2. Parameters for Set 1

qpec_type	cond_P(scale_P)	convex_f	symm_M	mono_M	cond_M (scale_M)		
100	100 (100)	1	1	1	200 (200)		
second_deg	first_deg	mix_deg	tol_deg	implicit	rand_seed	output	
0	2	0	1.e-6	0	0	3	

variable y , and the second level degeneracy does not exist at the generated solution. The starting point is chosen as $(100(z_1 - z_2), 1, \dots, 1) \in \mathfrak{R}^{n+m+p}$ where $z_1 \in \mathfrak{R}^{n+m}$ and $z_2 \in \mathfrak{R}^{n+m}$ are two randomly generated vectors with each component of z_1 and z_2 being uniformly distributed in $[0, 1]$.

Set 2, Problems 5–8. The parameters for these QPEC are the same as in Set 1 except for `second_deg` = 4 and `mix_deg` = 2. The same starting points are used as in Set 1.

Set 3, Problems 9–12. The parameters for these QPEC are the same as in Set 2 except for `mono_M` = 0 and `symm_M` = 0, i.e. the lower-level M matrix is not necessarily monotone or symmetric. We aim to check how monotonicity of the lower-level problem affects the numerical performance of different algorithms. The same starting points are used as in Set 1.

Set 4, Problems 13–16. The parameters for these QPEC are the same as in Set 2 except for `second_deg` = 8. The same starting points are used as in Set 1. Higher degeneracy of the lower-level problem makes PSQP harder from the viewpoint of optimality conditions and the decomposition theory, since the cardinality of the index set $\mathcal{A}(w^*)$ at a solution w^* increases exponentially in the degree of second-level degeneracy. It is certainly of interest to see how this property affects numerical performance of other algorithms.

Results for the methods NLP/`constr`, NLP/MINOS, PSQP and PIPA on the generated test problems using the first four sets of parameters are shown in Tables 4, 5, 6 and 7. The notation used in these tables is explained in Table 3. There are several failures in these tables. Among them “F1” and “F2” indicate that the final point is infeasible due to inability of the phase I procedure to find a feasible starting point for PSQP and PIPA respectively.

We do not report in Table 5 that PIPA suffers from a kind of failure on problems 3, 4, 9, 11, 13, 14, 16; that is, termination occurs because of a linesearch failure in Step 4 (the stepsize τ or χ_k becomes unacceptably small) rather than according to the stopping criteria in Step 2. This affects the accuracy of the solution found by PIPA.

Table 3. Notation used to describe numerical performance

(m, n, l, p)	Dimensions of the problem.
<code>second_deg</code>	# of degenerate lower-level indices at the generated solution $(x_{\text{gen}}, y_{\text{gen}})$.
<code>Iter</code>	# of iterations required by either PSQP, PIPA, or NLP via <code>constr</code> .
<code>Iter major</code>	# of linearly-constrained NLPs solved by MINOS.
<code>Iter minor</code>	# of pivots performed by MINOS.
<code>QPs to sol.</code>	# of quadratic programs used after reaching the found solution in PSQP.
<code>QPs total</code>	# of quadratic programs used after the termination of PSQP.
<code>Flops Phase I</code>	# of flops used in the Phase I in either PSQP or PIPA.
<code>Flops to sol.</code>	# of flops used after reaching the found solution in PSQP.
<code>Flops total</code>	# of flops used after the termination of either PSQP or PIPA or NLP.
<code>f</code>	The objective function value at the found solution.
<code>fgen</code>	The objective function value at the generated solution $(x_{\text{gen}}, y_{\text{gen}})$.
<code>Norm 1</code>	The infinity norm of the difference vector between the starting point and the generated solution $(x_{\text{gen}}, y_{\text{gen}})$.
<code>Norm 2</code>	The infinity norm of the difference vector between the found solution and the generated solution $(x_{\text{gen}}, y_{\text{gen}})$.

We make some comments for MINOS results in Table 7. A major iteration consists of solving a quadratic program (linearly constrained nonlinear program in general) by a sequence of fast minor iterations which involve a quasi-Newton approximation of the Hessian matrix and a generalization of the simplex method from linear programming to find a search direction. Also, we are not able to report flops in Table 7 because we used a MEX-file version of MINOS, which links the original Fortran code [24] to MATLAB.

5.2.3. Implementation of implicit QPEC: LCP-QP Recall that an MPEC is called an implicit program if its upper-level constraints are determined by the first-level variables. Implicit QPEC are an important class of MPEC. Unlike the previous two subsections, we consider LCP-QP instead of AVI-QP in this and the next subsections. We are interested both in seeing how different methods perform for implicit LCP-QP problems, and particularly to compare PIPA with the implicit version i-PIPA, our implementation of the PIPA given in the monograph [19] described below.

From our experiments, we found that neither `constr` nor MINOS performed well for our relatively large fifth set of QPEC, namely implicit LCP-QP problems. These nonlinear programming methods are not only very slow but often terminate at either a worse solution than the generated solution or at an infeasible point. Therefore, we

Table 4. Numerical results for PSQP on AVI-QP

Prob./ stop. rule	(m,n,l,p)	second_deg	Iter	QPs to sol./ total	Flops Phase I/ Flops to sol./ Flops total	f/ fgen	Norm 1 / Norm 2
1/A	(20,8,4,8)	0	2	2/3	3.1e+06/3.7e06/4.9e+06	-64.3065/-65.0099	82/0.110555
1/B	(20,8,4,8)	0	2	2/3	3.1e+06/3.7e06/4.9e+06	-64.3065/-65.0099	82/0.110555
2/A	(30,12,8,12)	0	4	4/5	1.5e+07/2.4e+07/2.7e+07	-118.887/-118.886	88/0.00428526
2/B	(30,12,8,12)	0	4	4/5	1.5e+07/2.4e+07/2.7e+07	-118.887/-118.886	88/0.00428526
3/A	(40,16,12,16)	0	5	5/6	4.0e+07/6.8e+07/7.4e+07	-74.1346/-74.1346	92/1.7e-15
3/B	(40,16,12,16)	0	5	5/6	4.0e+07/6.8e+07/7.4e+07	-74.1346/-74.1346	92/1.7e-15
4/A	(50,20,16,20)	0	5	5/6	9.7e+07/1.7e+08/2.0e+08	-133.579/-133.567	82/0.0164943
4/B	(50,20,16,20)	0	5	5/6	9.7e+07/1.7e+08/2.0e+08	-133.579/-133.567	82/0.0164943
5/A	(20,8,4,8)	4	3	3/19	3.5e+06/5.2e+06/1.7e+07	-71.4787/-71.4787	81/3.1e-15
5/B	(20,8,4,8)	4	3	3/4	3.5e+06/5.2e+06/5.9e+06	-71.4787/-71.4787	81/3.1e-15
6/A	(30,12,8,12)	4	3	3/19	1.9e+07/2.4e+07/6.0e+07	-115.8/-115.8	88/1.4e-15
6/B	(30,12,8,12)	4	3	3/7	1.9e+07/2.4e+07/3.2e+07	-115.8/-115.8	88/1.4e-15
7/A	(40,16,12,16)	4	4	4/20	4.3e+07/6.9e+07/2.8e+08	-50.5499/-50.5499	92/1.4e-14
7/B	(40,16,12,16)	4	4	4/7	4.3e+07/6.9e+07/1.1e+08	-50.5499/-50.5499	92/1.4e-14
8/A	(50,20,16,20)	4	4	4/20	8.7e+07/2.3e+08/1.9e+09	-73.1951/-73.1951	82/1.8e-15
8/B	(50,20,16,20)	4	4	4/5	8.7e+07/2.3e+08/2.5e+08	-73.1951/-73.1951	82/1.8e-15
9/A	(20,8,4,8)	4	3	3/7	3.6e+06/4.9e+06/9.9e+06	-82.6728/-83.9831	88/0.149311
9/B	(20,8,4,8)	4	3	3/7	3.6e+06/4.9e+06/9.9e+06	-82.6728/-83.9831	88/0.149311
10/A	(30,12,8,12)	4	2	2/10	1.9e+07/2.5e+07/6.2e+07	4133.13/-179.695	85/6.93468
10/B	(30,12,8,12)	4	2	2/5	1.9e+07/2.5e+07/3.8e+07	4133.13/-179.695	85/6.93468
11/A	(40,16,12,16)	4	2	F1	F1	F1	F1
11/B	(40,16,12,16)	4	2	F1	F1	F1	F1
12/A	(50,20,16,20)	4	4	4/20	1.4e+09/1.5e+09/1.9e+09	-110.073/-110.073	87/1.8e-15
12/B	(50,20,16,20)	4	4	4/6	1.4e+09/1.5e+09/1.5e+09	-110.073/-110.073	87/1.8e-15
13/A	(20,8,4,8)	8	2	2/258	1.3e+08/1.3e+08/4.9e+08	-87.0353/-87.0353	80/2.1e-15
13/B	(20,8,4,8)	8	2	2/19	1.3e+08/1.3e+08/1.5e+08	-87.0353/-87.0353	80/2.1e-15
14/A	(30,12,8,12)	8	2	2/258	1.4e+07/1.7e+07/6.8e+08	-128.801/-128.801	88/1.8e-15
14/B	(30,12,8,12)	8	2	2/4	1.4e+07/1.7e+07/2.2e+07	-128.801/-128.801	88/1.8e-15
15/A	(40,16,12,16)	8	3	3/259	4.4e+07/6.3e+07/3.2e+09	-86.3114/-86.3114	91/4.6e-15
15/B	(40,16,12,16)	8	3	3/54	4.4e+07/6.3e+07/6.9e+08	-86.3114/-86.3114	91/4.6e-15
16/A	(50,20,16,20)	8	4	4/260	8.5e+07/1.6e+08/6.7e+09	-47.3336/-47.3336	83/1.0e-14
16/B	(50,20,16,20)	8	4	4/7	8.5e+07/1.6e+08/2.3e+08	-47.3336/-47.3336	83/1.0e-14

Table 5. Numerical results for PIPA on AVI-QP

Problem	(m,n,l,p)	second_deg	Iter	Flops Phase I/ Flops total	f/ fgen	Norm 1 / Norm 2
1	(20,8,4,8)	0	9	3.1e+06/3.3e+07	-65.0099/-65.0099	82/7.4e-09
2	(30,12,8,12)	0	13	1.5e+07/1.5e+08	-118.887/-118.886	88/0.00428526
3	(40,16,12,16)	0	30	4.0e+07/5.3e+08	-74.1198/-74.1346	92/0.0230878
4	(50,20,16,20)	0	20	1.0e+08/9.8e+08	-133.558/-133.567	82/0.0236974
5	(20,8,4,8)	4	23	3.5e+06/7.3e+07	-71.4787/-71.4787	81/8.6e-09
6	(30,12,8,12)	4	25	1.9e+07/2.9e+08	-115.8/-115.8	88/6.2e-09
7	(40,16,12,16)	4	22	4.3e+07/6.4e+08	-50.5499/-50.5499	92/6.0e-09
8	(50,20,16,20)	4	22	8.5e+07/7.9e+08	-73.1951/-73.1951	82/6.2e-09
9	(20,8,4,8)	4	11	3.6e+06/4.3e+07	-82.6727/-83.9831	88/0.149313
10	(30,12,8,12)	4	27	3.9e+07/3.2e+08	4133.13/-179.695	85/6.93468
11	(40,16,12,16)	4	18	7.7e+08/1.2e+09	-71.7339/-71.7339	92/3.5e-05
12	(50,20,16,20)	4	F2	F2	F2	F2
13	(20,8,4,8)	8	28	4.8e+06/1.0e+08	-87.0353/-87.0353	80/2.3e-08
14	(30,12,8,12)	8	9	1.4e+07/1.2e+08	-128.801/-128.801	88/5.1e-05
15	(40,16,12,16)	8	28	4.0e+07/7.3e+08	-86.3114/-86.3114	91/3.3e-09
16	(50,20,16,20)	8	20	8.5e+07/1.2e+09	-47.3336/-47.3336	83/2.2e-07

Table 6. Numerical results for NLP using `constr` on AVI-QP

Problem	(m,n,l,p)	<code>second_deg</code>	Iter	Flops total	f/ fgen	Norm 1 / Norm 2
1	(20,8,4,8)	0	5	1.0e+07	2.72707e+06/-65.0099	82/236.57
2	(30,12,8,12)	0	9	8.9e+07	2.18626e+06/-118.886	88/167.237
3	(40,16,12,16)	0	72	1.7e+09	-74.1346/-74.1346	92/2.5e-05
4	(50,20,16,20)	0	36	2.1e+09	-132.356/-133.567	88/0.175634
5	(20,8,4,8)	4	12	2.8e+07	969283/-71.4787	81/154.543
6	(30,12,8,12)	4	11	1.2e+08	27914.7/-115.8	88/25.9409
7	(40,16,12,16)	4	51	1.3e+09	-50.4763/-50.5499	92/0.063909
8	(50,20,16,20)	4	52	3.3e+09	-68.5143/-73.1951	82/0.313929
9	(20,8,4,8)	4	11	2.8e+07	3182.52/-83.9831	88/7.5299
10	(30,12,8,12)	4	35	3.7e+08	-159.28/-179.695	85/0.50267
11	(40,16,12,16)	4	31	1.0e+09	-69.3414/-71.7339	92/0.287154
12	(50,20,16,20)	4	29	1.9e+09	165.839/-110.073	87/1.28856
13	(20,8,4,8)	8	11	2.7e+07	6.8e+06/-87.0353	80/335.317
14	(30,12,8,12)	8	56	5.2e+08	-128.751/-128.801	88/0.0156657
15	(40,16,12,16)	8	36	9.3e+08	-86.3114/-86.3114	91/0.000158901
16	(50,20,16,20)	8	34	2.1e+09	-38.595/-47.3336	83/0.320972

Table 7. Numerical results for NLP using MINOS on AVI-QP

Problem	(m,n,l,p)	second_deg	Iter major/ Iter minor	f/ fgen	Norm 1 / Norm 2
1	(20,8,4,8)	0	12/165	-65.0099/-65.0099	82/1.2e-12
2	(30,12,8,12)	0	19/197	-118.887/-118.886	88/0.00428526
3	(40,16,12,16)	0	8/147	-73.4631/-74.1346	92/0.125276
4	(50,20,16,20)	0	17/292	-132.545/-133.567	82/0.147682
5	(20,8,4,8)	4	11/143	-71.4787/-71.4787	81/3.4e-12
6	(30,12,8,12)	4	9/136	-115.8/-115.8	88/1.5e-13
7	(40,16,12,16)	4	9/136	-50.0828/-50.5499	92/0.156281
8	(50,20,16,20)	4	31/281	-73.171/-73.1951	82/0.0211662
9	(20,8,4,8)	4	11/72	-80.161/-83.9831	88/0.166252
10	(30,12,8,12)	4	21/215	-179.695/-179.695	85/4.5e-11
11	(40,16,12,16)	4	10/209	-68.1702/-71.7339	92/0.342033
12	(50,20,16,20)	4	4/178	-61.7592/-110.073	87/0.603738
13	(20,8,4,8)	8	10/72	-87.0353/-87.0353	80/5.1e-13
14	(30,12,8,12)	8	17/130	-128.801/-128.801	88/3.8e-13
15	(40,16,12,16)	8	15/169	-86.3114/-86.3114	91/4.1e-10
16	(50,20,16,20)	8	15/309	-47.3336/-47.3336	83/8.7e-13

do not report numerical behavior of `constr` and `MINOS` for our implicit LCP-QP problems.

Piecewise Sequential Quadratic Programming Method

The parameters used are set as in Subsubsection 5.2.1. Since LCP-constrained implicit programs are considered in this subsection, we give the following approximate decomposition: given a point $z^* = (x^*, y^*)$, define

$$\begin{aligned}\alpha(z^*, \delta) &= \{1 \leq i \leq m : y_i^* - (Nx^* + My^* + q)_i > \delta\} \\ \beta(z^*, \delta) &= \{1 \leq i \leq m : |y_i^* - (Nx^* + My^* + q)_i| \leq \delta\} \\ \gamma(z^*, \delta) &= \{1 \leq i \leq m : (Nx^* + My^* + q)_i - y_i^* > \delta\}\end{aligned}$$

and the family of index sets

$$\mathcal{A}(z^*, \delta) = \{\alpha \subseteq \{1, \dots, m\} : \alpha \supseteq \alpha(z^*, \delta), \alpha^c \supseteq \gamma(z^*, \delta)\},$$

where α^c is the complement set of α with respect to $\{1, \dots, m\}$.

In phase I, after choosing a point x^0 satisfying the upper level constraints $Gx + a \leq 0$, Lemke's method is used to find y^0 such that (x^0, y^0) is feasible, by solving the lower-level LCP for this fixed x^0 .

The main M-file used to implement PSQP for the implicit LCP-QP is `psqp1cp`, which also requires three M-files `f`, `df` and `d2f`. The MATLAB QP solver `qp` is used in Step 1 of PSQP. Only the version of PSQP with Stopping rule B is tested since this is clearly superior to using Stopping rule A.

Penalty Interior-Point Algorithm

The parameters used are chosen as in Subsubsection 5.2.1. But, Q_k is chosen as follows,

$$Q_k = \begin{pmatrix} \nabla^2 f(z^k) & 0 \\ 0 & \mu_k I \end{pmatrix}$$

where I is the identity matrix of dimension m .

In phase I, Lemke's method is used to find a strictly feasible starting point (x^0, y^0, w^0) by first choosing a point x^0 that satisfies the upper-level constraints $Gx + a \leq 0$, then determining a solution y of the following modified LCP

$$0 \geq y - My + (Nx^0 + Me + q - e) \geq 0,$$

and, finally, letting $y^0 = y + e$ and $w^0 = Nx^0 + My^0 + q$.

The main M-file `pipalcp` needs the M-files `f`, `df`, `d2f` and `pipapenlcp` whose purposes should be clear from previous discussion. The quadratic program in Step 1 is solved by the MATLAB solver `qp`.

Implicit Penalty Interior-Point Algorithm, i-PIPA

If the LCP-QP is an implicit program and M is monotone or of some similar property, then in the corresponding quadratic program used to generate a search direction in the PIPA, other variables can be uniquely determined by dx . This leads to solving a smaller dimension quadratic program in the variables dx , which

obviously saves time if m is large and the matrix data M , N in LCP-QP of the form (5) are dense. In fact, the version of PIPA implemented in [19] takes full advantage of this idea. We have also coded a special version of PIPA called i-PIPA for implicit LCP-constrained quadratic programs, which uses this dimension reduction strategy and allows for infeasible but interior starting points, and tested it in Problem Set 5. The prefix “i” in i-PIPA stands for “implicit”. The difference between i-PIPA and the version of PIPA implemented in [19] lies in that i-PIPA does not use the simple bounds which are used in [19] for controlling the magnitude of remaining components $(dy, d\lambda, dv)$.

The parameters used in i-PIPA are set as in Subsubsection 5.2.1 with the one exception that i-PIPA terminates if $\|(dx, dy, dw)\|_\infty \leq \varepsilon_1$ and $\mu_k \leq \varepsilon_2$, where $\varepsilon_1 = 10^{-4}$. We use a bigger ε_1 here than previously because this allows i-PIPA to terminate much more quickly with a relatively small loss of accuracy.

5.2.4. Numerical results for implicit QPEC: LCP-QP

Set 5, Problems 17–24. This set of test problems is designed for testing implicit programs LCP-QP rather than AVI-QP problems in the first three sets. The parameters for this set of problems are the same as in Set 2 except for `implicit=1` and `qpec.type = 300`. The same starting points are used for i-PIPA as in Set 1. However, in phase I of PSQP and i-PIPA, these starting points are not used in the code. Instead, Lemke’s method is used to find a good starting point for both PSQP and PIPA. Some details can be found in the previous subsection.

Results for the methods PSQP and i-PIPA on the fifth set implicit LCP-QP problems are reported in Table 8. The meaning of notation used in this table is the same as in Table 3.

5.3. Discussion of Numerical Results

From the numerical results, we have the following impressions.

- The naive nonlinear programming approach has more difficulties than other methods as expected. Though MINOS performs respectably, it has more trouble locating the generated solution w_{gen} , or a better feasible point, than do PSQP and PIPA. The NLP approach implemented using `constr` has considerable difficulty in finding w_{gen} , or a better point, and could not be recommended on this basis.

Also, a quirk of using MINOS in MATLAB is that consecutive runs on the same problem in one MATLAB session can produce different solutions, though closing and restarting MATLAB after each run of MINOS gives the same results. So we obtained the results in Table 7 by closing and restarting MATLAB after each run of MINOS. (For Problem 4 of Set 1, to avoid crashing MATLAB we were obliged to run MINOS on another problem before testing it on Problem 4.)

Table 8. Numerical results for implicit LCP-QP problems

Problem / Algorithm	(m,n,l)	second_deg	Iter	QPs total	Flops Phase I/ Flops total /	f/ fgen	Norm 1 / Norm 2
17/PSQP/B	(50,8,4)	0	4	4	716487/6.8e+06	-142.829/-142.829	2.2/1.1e-15
17/PIPA			24	24	23549/9.3e+08	-142.829/-142.829	2.2/1.6e-08
17/i-PIPA			27	27	(N/A)/1.1e+07	-142.829/-142.829	100/1.0e-07
18/PSQP/B	(100,8,4)	0	5	5	1.3e+07/9.3e+07	-664.389/-664.389	1.9/3.9e-15
18/PIPA			23	23	113526/6.9e+09	-664.389/-664.389	1.9/1.2e-08
18/i-PIPA			26	26	(N/A)/5.9e+07	-664.389/-664.389	100/1.7e-07
19/PSQP/B	(150,8,4)	0	7	7	7.6e+07/5.3e+08	-535.743/-535.743	2.6/9.0e-15
19/PIPA			25	25	266007/2.3e+10	-535.743/-535.743	2.6/2.4e-08
19/i-PIPA			26	26	(N/A)/1.7e+09	-535.743/-535.743	100/3.7e-06
20/PSQP/B	(200,8,4)	0	31	31	2.4e+07/7.3e+08	-109.595/-109.595	3.0/4.5e-15
20/PIPA			27	27	432915/5.8e+10	-109.595/-109.595	3.0/1.7e-08
20/i-PIPA			28	28	(N/A)/4.0e+08	-109.595/-109.595	100/2.3e-06
21/PSQP/B	(50,8,4)	4	4	4	468822/7.3e+06	41.8764/-41.8764	2.3/5.8e-15
21/PIPA			44	44	25351/1.7e+09	-41.8764/-41.8764	2.3/5.6e-09
21/i-PIPA			33	33	(N/A)/1.4e+07	-41.8764/-41.8764	100/1.0e-04
22/PSQP/B	(100,8,4)	4	6	6	1.2e+07/1.2e+08	-599.936/-599.936	2.1004/2.9e-14
22/PIPA			45	45	106430/9.9e+09	-599.936/-599.936	2.1004/9.5e-09
22/i-PIPA			53	53	(N/A)/1.2e+08	-599.936/-599.936	100/8.4e-5
23/PSQP/B	(150,8,4)	4	8	8	7.2e+07/5.7e+08	-536.444/-536.444	3.5146/3.5e-14
23/PIPA			43	43	281518/3.0e+10	-536.444/-536.444	3.5146/3.1e-08
23/i-PIPA			60	60	(N/A)/3.9e+08	-536.444/-536.444	100/5.0e-05
24/PSQP/B	(200,8,4)	4	28	28	1.9e+07/7.1e+08	-23.7817/-23.7817	2.38818/3.5e-15
24/PIPA			46	46	345537/9.5e+10	-23.7817/-23.7817	2.38818/4.8e-09
24/i-PIPA			73	73	(N/A)/1.0e+09	-23.7817/-23.7817	100/9.9e-05

In addition, MINOS was associated with occasional crashes of MATLAB, a problem that we believe is associated with the MEX interface to MATLAB rather than with MINOS itself.

- PSQP/B, i.e. PSQP with stopping condition B, appears to be the most effective method in terms of flops (compared to PSQP/A and PIPA) and accuracy (compared to all other methods) for problems with monotone equilibrium constraints, Sets 2 and 4. However we believe that the apparent advantage of PSQP/B, at least relative to PIPA, may not extend to QPEC of large dimension or nonlinear problems. Moreover the accuracy measure — the norm of the distance between the final (x, y) pair and the generated solution $(x_{\text{gen}}, y_{\text{gen}})$ — is most useful when the best solution found coincides with $(x_{\text{gen}}, y_{\text{gen}})$, which is not always the case, e.g. for Problems 2 and 4, some methods find better solutions than $(x_{\text{gen}}, y_{\text{gen}})$. Further investigation is certainly required.
- Nondegenerate problems, Set 1, were generally a little easier for PIPA in terms of numbers of iterations and flops than degenerate problems in later test sets. Naturally PSQP/A performed identically to PSQP/B on nondegenerate problems, Set 1; however PSQP/A was significantly disadvantaged by (second-level) degeneracy, though the implementation using stopping rule B was only mildly affected, even for problems with a relatively high degree of degeneracy, Set 4. It is unclear whether the degree of degeneracy had any effect on the NLP solvers `unconstr` and MINOS.
- For problems with nonmonotone equilibrium constraints, Set 3, it is not clear from our results if any algorithm dominates the others. For instance, both PSQP and PIPA fail to find a feasible starting point on a problem in Set 3, though this may not truly constitute a disadvantage for PIPA given that an infeasible-point implementation is possible as indicated previously. Also in Set 3, most algorithms experience more difficulty in locating w_{gen} than for problems with monotone equilibrium constraints.
- We did carry out several tests on problems generated by `QPECgen` for which the parameters match Table 2 except that `cond_P` and `cond_M` are increased. These results are not reported because until the condition number is very large, say 10^{18} or greater, the performance of the above methods is similar to the results above. For very large condition numbers, all methods experienced difficulties as expected. We suspect that more modest levels of ill-conditioning will play a greater role for nonconvex and more particularly, large-scale problems.
- We have not presented any results for the case when the objective function is not convex, problems where the dimension is large, or ill-conditioning features. Larger problems proved to be prohibitive in terms of memory requirements on the machine used for numerical tests. This suggests that future work requires routines, specifically a QP solver, designed for large and sparse problems.

Table 9. Comparisons of algorithms for AVI-QP problems

Algorithm	# reach $(x_{\text{gen}}, y_{\text{gen}})$ (accurately)		# reach best solution	# success	# QPs	Wins in Flops
PSQP/A	10	(10)	13	15	1131	4
PSQP/B	10	(10)	13	15	145	14
PIPA	10	(6)	11	15	305	1
constr	2	(0)	2	16	491	2
MINOS	8	(8)	9	16	219	N/A

It is to be expected that in each of these cases, the above methods will experience more difficulties than for the current test sets. However these issues will need to be explored in future work.

- The implicit LCP-QP results in Table 8 show that i-PIPA is more efficient in terms of flops than PIPA for these problems. In other respects, these results are similar to previous results on Test Set 2.

To make some comparisons between the different algorithms for all four sets of AVI-QP test problems, 16 problems in all, we give one more table, Table 9. In this table, five measurements are presented. They are “# reach $(x_{\text{gen}}, y_{\text{gen}})$ ” or the number of times each particular method finds the generated solution, by which we mean the number of times that the (infinity-norm) distance between the final point and $(x_{\text{gen}}, y_{\text{gen}})$ is not greater than 10^{-3} ; “# reach $(x_{\text{gen}}, y_{\text{gen}})$ accurately”, the number of times that the distance is not greater than 10^{-8} ; “# reach best solution” which is the the number of times each method converges to the best found solution, which in these tests may actually be worse than the generated solution but not better; “# success”, the number of AVI-QP test problems where the method concerned indicated successful termination, not necessarily termination at the generated solution or the best solution found (including runs of PIPA that terminated due to line search difficulties as previously mentioned); “# QPs”, the total number of quadratic programs solved over all the successfully terminated runs; and finally “Wins in Flops”, the number of times each method was better than all other methods in terms of flops for the successfully terminated runs, irrespective of whether different solutions were arrived at.

Table 9 seems to recommend PSQP/B above other methods. However, in light of the comments at the start of this section on global vs local minima, we believe Table 9 indicates that the problems tested have few local minimizers. Our conclusion overall is that PSQP/B is very efficient for small to medium size QPECs under the provision that few local minimizers exist. More general conclusions may still be somewhat premature in the field of MPEC.

6. Final Remarks

In this paper, we have developed a technique for generating different QPEC problems with certain important options under the control of the user. We have also implemented and tested several algorithms for solving QPEC within MATLAB. Although the relative merits and demerits of each method cannot be completely determined from our limited tests, the problems generated by `QPECgen` do give rise to recognizably different algorithm behaviours.

It has been recognized that `qp` in MATLAB is not fully satisfactory when the quadratic program is not positive definite. Therefore, one part of our future work to implement the various methods using more robust quadratic programming solvers. We also believe that some of the most important computational issues to be faced involve nonlinear and large-scale problems.

Acknowledgments

This work is supported by the Australian Research Council. We thank anonymous referees for their comments. We also thank Michael Ferris for his interest in the project and helping us to use MINOS in MATLAB; Jong-Shi Pang for comments and discussions; and Francis Tin-Loi for his encouragement and feedback on applying a previous PSQP code.

References

1. E. Aiyoshi and K. Shimizu, "Hierarchical decentralized systems and its new solution by a barrier method," *IEEE Transactions on Systems, Man, and Cybernetics SMC-11* pp. 444–449, 1981.
2. G. Anandalingam and T.L. Friesz, eds., "Hierarchical Optimization," *Annals of Operations Research*, 1992.
3. Z. Bi, P. Calamai and A. Conn, "An exact penalty function approach for the nonlinear bilevel programming problem," Technical Report #180-O-170591, Department of Systems Design Engineering, University of Waterloo, 1991.
4. P. Calamai and L.N. Vicente, "Generating quadratic bilevel programming problems," *ACM Transactions on Mathematical Software* vol. 20 pp. 103–122, 1994.
5. Y. Chen and M. Florian, "The nonlinear bilevel programming problem: formulations, regularity and optimality conditions," *Optimization* vol. 32 pp. 193–209, 1994.
6. R.W. Cottle, J.S. Pang and R.E. Stone, *The Linear Complementarity Problem*, Academic Press: New York, 1992.
7. S. Dempe, "On generalized differentiability of optimal solutions and its application to an algorithm for solving bilevel optimization problems," in: D. Du, L. Qi and R. Womersley, eds., *Recent Advances in Nonsmooth Optimization*, World Scientific Publishers, Singapore, 1995, pp. 36–56.
8. S. Dirkse and M.C. Ferris, "Modeling and solution environments for MPEC: GAMS & MATLAB, technical report," Computer Sciences Department, University of Wisconsin–Madison, Madison, 1997.
9. F. Facchinei, H. Jiang and L. Qi, "A smoothing method for mathematical programs with equilibrium constraints," *Mathematical Programming*, to appear.
10. J.E. Falk and J. Liu, "On bilevel programming, Part I: general nonlinear cases" *Mathematical Programming* vol. 70 pp. 47–72, 1995.
11. R. Fletcher, *Practical Methods of Optimization*, John Wiley & Sons: New York, second edition, 1987.
12. T. Friesz, R. Tobin, H. Cho and N. Mehta, "Sensitivity analysis based heuristic algorithms for mathematical programs with variational inequality constraints," *Mathematical Programming* vol. 48 pp. 265–284, 1990.
13. M. Fukushima and J.S. Pang, "Some feasibility issues in mathematical programs with equilibrium constraints," *SIAM Journal on Optimization*, to appear.
14. M. Fukushima, Z.-Q. Luo and J.S. Pang, "A globally convergent sequential quadratic programming algorithm for mathematical programs with linear complementarity constraints," *Computational Optimization and Applications*, to appear.
15. G.H. Golub and C.F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press: Baltimore, 1983.
16. J.Y. Han, G. Liu and S. Wang, "A new descent algorithm for solving quadratic bilevel programming problems," Manuscript, Institute of Applied Mathematics, Chinese Academy of Science, Beijing, 1997.
17. B. Hobbs, C. Metzler and J.S. Pang, "Strategic gaming analysis for electric power networks: an MPEC approach," Report, Department of Mathematical Sciences, The Johns Hopkins University, forthcoming.
18. M. Kočvara and J.V. Outrata, "On the solution of optimum design problems with variational inequalities," in: D.Z. Du, L. Qi and R.S. Womersley, eds., *Recent Advances in Nonsmooth Optimization*, World Scientific Publishers, Singapore, 1995, pp. 172–192.
19. Z.-Q. Luo, J.S. Pang and D. Ralph, *Mathematical Programs with Equilibrium Constraints*, Cambridge University Press: New York, 1997.
20. Z.-Q. Luo, J.S. Pang and D. Ralph, "Piecewise sequential quadratic programming for mathematical Programs with nonlinear complementarity constraints," in: A. Migdalas *et al*, eds., *Multilevel Optimization: Algorithms, Complexity and Applications*, Kluwer Academic Publishers, 1998, pp. 209–229.
21. Z.-Q. Luo, J.S. Pang, D. Ralph and S.-Q. Wu, "Exact penalization and stationarity conditions of mathematical programs with equilibrium constraints," *Mathematical Programming* vol. 75 pp. 19–76, 1996.

22. MATLAB 4.2, The MathWorks, Inc.: 24 Prime Park Way, Natick MA, 1994.
23. A. Migdalas and P.M. Pardalos, "Nonlinear bilevel problems with convex second level problem – Heuristics and descent methods," in: D.Z. Du, X.S. Zhang and K. Chuan, Operations Research and Its Applications, World Publishing Corporation, Singapore, 1995, pp. 194–204.
24. B.A. Murtagh and M.A. Saunders, "MINOS 5.4 user's guide," Technical Report SOL 83-20R, Systems Optimization Laboratory, Stanford University, CA, 1983.
25. J.V. Outrata, "On optimization problems with variational inequality constraints," SIAM Journal on Optimization vol. 4 pp. 340–357, 1994.
26. J.V. Outrata and J. Zowe, "A numerical approach to optimization problems with variational inequality constraints," Mathematical Programming vol. 68 pp. 105–130, 1995.
27. J.S. Pang, "Complementarity problems," in: R. Horst and P. Pardalos, eds., Handbook of Global Optimization, Kluwer Academic Publishers, Boston, 1995, pp. 271–338.
28. D. Ralph, "Sequential quadratic programming for mathematical programs with linear complementarity constraints," in: R.L. May and A.K. Easton eds., Computational Techniques and Applications: CTAC95, World Scientific, 1996, pp. 663–668.
29. H. Scheel and S. Scholtes, "Mathematical programs with equilibrium constraints: Stationarity, optimality, and sensitivity," manuscript, Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, England, 1996.
30. S. Scholtes and M. Stöhr, "Exact penalization of mathematical programs with equilibrium constraints," Report 26, The Judge Institute of Management Studies, University of Cambridge, England 1997.
31. L.N. Vicente and P. Calamai, "Bilevel and multilevel programming: A bibliography review," Journal of Global Optimization vol. 5 pp. 291–306, 1994.
32. L. Vicente, G. Savard and J. Júdice, "Descent approaches for quadratic bilevel programming," Journal of Optimization Theory and Applications vol. 81 pp. 379–399, 1994.