Projection Based Algorithms for Variational Inequalities

Sudhanshu Shekhar Singh

A thesis submitted to the faculty of the University of North Carolina at Chapel Hill in fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Statistics and Operations Research.

Chapel Hill
2010

Approved by

Shu Lu

Jon Tolle

David Rubin

Scott Provan

Gabor Pataki

Abstract

SUDHANSHU SHEKHAR SINGH: Projection Based Algorithms for Variational Inequalities
(Under the supervision of Dr. Shu lu)

This dissertation is about the theory and iterative algorithms for solving variational inequalities. Chapter 1 introduces the problem, various situations in which variational inequalities arise naturally, reformulations of the problem, several characteristics of the problem based on those reformulations, as well as the basic existence and uniqueness results. Following that, chapter 2 describes the general approaches to solving variational inequalities, focusing on projection based methods towards the end, with some convergence results. That chapter also discusses the merits and demerits of those approaches. In chapter 3, we describe a relaxed projection method, and a descent method for solving variational inequalities with some examples. An application of the descent framework to a game theory problem leads to an algorithm for solving box constrained variational inequalities. Relaxed projection methods require a sequence of parameters that approach zero, which leads to slow convergence as the iterates approach a solution. Chapter 4 describes a local convergence result that can be used as a guideline for finding a bound on the parameter as a relaxed projection algorithm reaches a solution.

Table of Contents

List of Algorithms

CHAPTER 1

Introduction

Given a non-empty, closed and convex subset K of $\mathbb{R}^n$ and a mapping F from K to $\mathbb{R}^n$, the variational inequality problem VI(K, F) is to find a vector $x^* \in K$ such that

$$\langle F(x^*), x - x^* \rangle \geq 0 \quad \forall\, x \in K. \tag{1.1}$$

The solution set of (1.1) is referred to as SOL(K, F). If F is continuous, then SOL(K, F) is closed.

Variational inequalities were introduced by Hartman and Stampacchia in 1966 for the study of partial differential equations with applications in the field of mechanics. Those variational inequalities were infinite dimensional. Finite dimensional VI theory was developed later in 1980 when Dafermos observed that traffic network equilibrium conditions have a structure of VI.

Variational inequalities provide a tool to formulate various equilibrium problems. Several well known problems, such as systems of nonlinear equations, first order conditions for linear and nonlinear optimization problems, and complementarity problems, are special cases of variational inequalities.

A geometric interpretation of a solution of VI(K, F) is that x^* is a solution if and only if $F(x^*)$ makes a non-obtuse angle with all the feasible directions going into K from x^*. Alternately, we may say that x^* is a solution of VI(K, F) if and only if $-F(x^*)$ is in the normal cone to K at x^*. The normal cone to K at $x \in K$ is defined as

$$N(K, x) = \{d \in \mathbb{R}^n \mid \langle d, y - x \rangle \leq 0, \quad \forall\, y \in K\}. \tag{1.2}$$

If a solution x^* of (1.1) belongs to the interior of K, then $F(x^*) = 0$. To see this, note that the

fact $x^* \in int(K)$ implies that $x^* - \tau F(x^*) \in int(K)$ for sufficiently small τ. Hence (1.1) implies $\langle F(x^*), -F(x^*) \rangle \geq 0$, that is $F(x^*) = 0$. In particular if $K = \mathbb{R}^n$, the solution set of (1.1) is the same as that of $F(x) = 0$.

1.1 Sources of variational inequalities

Complementarity problems

Variational inequalities are very closely related to complementarity problems (CP). When K is a cone, the VI assumes the form of a CP. For completeness, we introduce the definition of a CP. Given a cone $K \subset \mathbb{R}^n$ and a mapping $F : K \to \mathbb{R}^n$, the complementarity problem, CP(K, F), is to find $x \in K$ such that $F(x) \in K^*$ and,

$$\langle x, F(x) \rangle \geq 0, \tag{1.3}$$

where K^* is the dual cone of K defined as,

$$K^* = \{y \in \mathbb{R}^n \mid \langle y, x \rangle \geq 0, \quad \forall\, x \in K\}. \tag{1.4}$$

Two special cases of CP are very important:

(a) The nonlinear complementarity problem NCP(F): Let $\mathbb{R}^n_+$ denote the nonnegative othant of $\mathbb{R}^n$, and let $F : \mathbb{R}^n \to \mathbb{R}^n$. The nonlinear complementarity problem (NCP) is to find $x^* \in \mathbb{R}^n_+$ such that

$$\begin{aligned} F(x^*) &\geq 0, \\ \langle F(x^*), x^* \rangle &= 0. \end{aligned} \tag{1.5}$$

The solution set to NCP and VI($\mathbb{R}^n_+, F$) are the same. The equivalence is easy to observe. If x^* solves the NCP, then

$$\langle F(x^*), x - x^* \rangle \geq 0, \quad \forall\, x \in \mathbb{R}^n_+, \tag{1.6}$$

because $F(x^*) \geq 0$, $x \in \mathbb{R}^n_+$ and $\langle F(x^*), x^* \rangle = 0$. On the other hand, if $x^* \in$ SOL($\mathbb{R}^n_+$, F), then (1.6) holds. Substituting $x = 0$ and $x = 2x^*$ in (1.6) implies $\langle F(x^*), x^* \rangle = 0$. Moreover, substituting $x = e_i$ in (1.6), where e_i is the i^{th} unit vector in $\mathbb{R}^n$, shows that $F(x^*) \geq 0$.

(b) The mixed complementarity problem MiCP(G, H): Let G and H be mappings from $\mathbb{R}^{n_1} \times \mathbb{R}^{n_2}_+$ to $\mathbb{R}^n_1$ and $\mathbb{R}^n_2$ respectively. The MiCP(G, H) is to find a pair of vectors (u, v) such that $v \geq 0$ and

$$G(u, v) = 0, \ \langle v, H(u, v) \rangle = 0. \tag{1.7}$$

System of equations

As already mentioned, solving $F(x) = 0$ is equivalent to solving VI($\mathbb{R}^n$, F). Hence any system of equations can be considered as a variational inequality problem.

Optimization problems

Variational inequalities arise from various sources, constrained optimization problems being one of those. Consider the optimization problem,

$$\min \ f(x), \quad s.t. \quad x \in K. \tag{1.8}$$

If x is a local solution to this optimization problem, then the gradient of f at x must make a non obtuse angle with all the feasible directions going into K from x. This is precisely the definition of the VI with $F = \nabla f$, that is, a local solution x of (1.8) satisfies

$$\langle y - x, \nabla f(x) \rangle \geq 0, \quad \forall \ y \in K. \tag{1.9}$$

But this does not imply that every VI can be rephrased as an optimization problem, because $F : \mathbb{R}^n \to \mathbb{R}^n$ is the gradient of some function if and only if the Jacobian of F is symmetric. Hence variational inequalities are more general than optimization problems.

Saddle Point problems

Let $L : \mathbb{R}^{n+m} \to \mathbb{R}$ be an arbitrary function and $X \subset \mathbb{R}^n$ and $Y \subset \mathbb{R}^m$ be two closed sets. The saddle point problem associated with (L, X, Y) is to find $(x, y) \in X \times Y$ such that

$$L(x, v) \leq L(x, y) \leq L(u, y), \quad \forall\ (u, v) \in \mathbf{X} \times \mathbf{Y}. \tag{1.10}$$

If L is convex on Y for a fixed x, and concave on X for a fixed y, then (x, y) solves the saddle point problem if and only if (x, y) solves the VI($X \times Y$, F) where

$$F = \begin{bmatrix} \nabla_u L(u, v) \\ -\nabla_v L(u, v) \end{bmatrix}.$$

Nash equilibrium problems

Certain Nash equilibrium problems can be solved as variational inequalities. For example, consider the multiplayer noncooperative game with N players. Let $K_i \in \mathbb{R}^{n_i}$ be the set of strategies that player i can employ. Here n_i is a positive integer. Each player has a cost function $\theta_i(x_1, x_2, ..., x_N)$ which depends on her own strategy and those of the other players. Given the strategies of the other players, player i wants to determine a strategy y_i that minimizes θ_i. So, the problem for player i is

$$\min\ \theta_i(x_i, x^i) \quad \text{subject to} \quad x_i \in K_i, \tag{1.11}$$

where $x^i = \{x_j\}_{j \neq i}$. Let $S_i(x)$ be the solution set to (1.11). The Nash equilibrium is achieved at a collection of strategies $x = \{x_i\}_{i=1}^N$ such that $x_i \in S_i(x)$.

If each K_i is convex, and for a fixed x^i, $\theta_i(x_i, x^i)$ is convex and continuously differentiable in x_i, then x is a Nash equilibrium if and only if it solves VI(K, F) with $K = \Pi_{i=1}^N K_i$ and $F = (\nabla_{x_i} \theta_i(x))_{i=1}^N$.

A specific instance of the above framework is the Nash Cournot production model (Harker, 1984) in which various players producing a single product have to determine an optimal quantity to

produce so that each player maximizes her own profit.

Economic equilibrium models

Many economic activity models can be formulated as VI/CP. We will briefly explain the Walrasian equilibrium model, which predicts equilibrium activities and prices in an economy for which interactions between the commodities comprising this economy have been incorporated. Consider an economy with m economic activities and n goods. The unit cost of operating activity i is c_i and the level of activity i is denoted by y_i. The initial endowment for good j is b_j, its unit price is denoted by p_j, and its demand function is $d_j(p)$ where p is the vector of prices of all the goods.

The relation between the activity levels and the availability of goods is given by a technology input-output matrix $A(p)$. A is an $m \times n$ matrix with entries $a_{ij}(p)$ such that $\langle A^T(p), y\rangle$ gives the vector of goods resulting from a vector y of activities; $\langle A(p), p\rangle$ is the vector of per unit activity returns. An activity price pair (y, p) is a general equilibrium if $(y, p) \geq 0$ and

$$\begin{aligned} c - \langle A(p), p\rangle &\geq 0 \\ \langle y, c - \langle A(p), p\rangle\rangle &= 0, \\ b + \langle A(p)^T, y\rangle - d(p) &\geq 0 \\ \langle p, b + \langle A(p)^T, y\rangle - d(p)\rangle &= 0. \end{aligned}$$

The first two conditions here mean that all activity levels are nonnegative, all activities give nonnegative profits, and activities with negative profits are not performed. The last two conditions state that the prices are non-negative, the supply must be at least as much as the demand, and supply exceeds demand only for free goods. These four conditions together define a complementarity problem.

Traffic equilibrium model

The traffic equilibrium model predicts the steady state traffic in a network in which all users are trying to use the network to minimize their travel cost. The sets of nodes and arcs of the network are denoted by N and A respectively. The flow on the network is given by a flow vector $f = \{f_a \mid a \in A\}$. The cost of using arc a when the flow in the network is f is given by a cost

function $c_a(f)$. There are two subsets, O and D, of N, that are recognized as origin and destination sets respectively. The set of all origin-destination (OD) pairs is a subset W of $O \times D$.

For an OD pair $w \in W$, let P_w be the set of all paths connecting w and let P be the union of all paths for all $w \in W$. Let h_p be the flow on path $p \in P$ and h be the vector of path flows. Let us denote the travel cost on path p as $C_p(h)$. Let Δ be the arc path incidence matrix whose (a, p) entry is 1 if arc $a \in A$ is on path $p \in P$, and 0 otherwise. For each $w \in W$, a function $d_w(u)$ represents the demand for the OD pair w as a function of u, the vector of minimum travel costs between all OD pairs.

The Wardrop user equilibrium principle postulates that users of the traffic network will choose minimum cost path between each OD pair, and through this process the paths that are used will have equal costs. Moreover, paths with costs higher than the minimum will have no flow. Also, the travel demand must be satisfied, and all the costs should be nonnegative. These conditions, written mathematically, state (Dafermos, 1980)

$$\begin{aligned} C_p(h) \leq 0, \quad h_p \geq 0 \\ \langle C_p(h) - u_w, h_p \rangle \geq 0, \ \forall\ w \in W_p \in P_w, \\ \sum_{p \in P_w} h_p = d_w(u), \ \forall\ w \in W, \\ u_w \geq 0, \ \forall\ w \in W. \end{aligned}$$

The static traffic user equilibrium problem is to find a pair (h, u) of path flows and minimum travel costs, so that the above conditions are satisfied. Under the assumption that the travel costs and demand functions are nonnegative, this is a complementarity problem.

CPs in symmetric PSD matrices

Let M^n denote the subspace of $\mathbb{R}^{n \times n}$ consisting of all symmetric $n \times n$ matrices. Let $M^n_+ \subset M^n$ be the cone of all positive semidefinite (PSD) $n \times n$ matrices. Like the nonnegative orthant of $\mathbb{R}^n$, M^n_+ is self dual under matrix multiplication. Let F be a mapping from M^n to M^n. The complementarity problem in SPSD matrices is to find a matrix $A \in M^n_+$ such that $F(A) \in M^n_+$ and

$$AF(A) \in M^n_+. \tag{1.12}$$

This problem is closely related to semidefinite programs in which the unkowns are symmetric PSD matrices.

There are various other applications of VI and CP like frictional contact problems, elastoplastic structural analysis problems, nonlinear obstacle problems etc. For a detailed description of these problems, the reader is referred to Chapter 1 of the book (Facchinei and Pang, 2003).

1.2 Equivalent formulations

We can obtain equivalent formulations of VI (and CP) as systems of equations, optimization problem or fixed point problem. These formulations are helpful in developing analytical results and algorithms for solving VIs.

Let us define the function F_{min} as

$$F_{min}(x) = min(x, F(x)) \tag{1.13}$$

where the $\min(\cdot, \cdot)$ is a componentwise minimum operator. We can rewrite (1.13) as

$$F_{min}(x) = x - \Pi_{\mathbb{R}^n_+}(0^n, x - F(x)), \tag{1.14}$$

where 0^n is the origin in $\mathbb{R}^n$ and $\Pi_{\mathbb{R}^n_+}()$ is the Euclidean projection onto $\mathbb{R}^n_+$. This formulation of $F_{min}(x)$ leads to two equation reformulations of the VI. Both formulations require the projection operator. Since the projection operator plays a very important role in the study of VI, it is imperative that we define it formally and mention some of its elementary properties (Facchinei and Pang, 2003).

The projection of a point $y \in \mathbb{R}^n$ on a closed convex subset K of $\mathbb{R}^n$, denoted by $\Pi_K(y)$, is the point $x \in K$ that is closest to y. The problem of finding $\Pi_K(y)$ is the following

$$\min \quad \frac{1}{2}\langle x - y, x - y\rangle, \quad \text{subject to } x \in K. \tag{1.15}$$

The problem (1.15) has a convex feasible set and a strongly convex objective function, so it has a

unique solution. Moreover

1. for each $x \in \mathbb{R}^n$, $\Pi_K(x)$ exists and is unique;

2. for each $x \in \mathbb{R}^n$, $\Pi_K(x)$ is the unique vector $y \in K$ satisfying

$$\langle z - y, y - x \rangle \geq 0, \quad \forall z \in K; \tag{1.16}$$

3. for any two vectors u and v in $\mathbb{R}^n$,

$$\langle \Pi_K(u) - \Pi_K(v), u - v \rangle \geq \|\Pi_K(u) - \Pi_K(v)\|_2^2; \tag{1.17}$$

4. $\Pi_K(x)$ is nonexpansive. That is, for any two vectors u and v in $\mathbb{R}^n$,

$$\|\Pi_K(u) - \Pi_K(v)\|_2 \leq \|u - v\|_2; \tag{1.18}$$

5. the squared distance function,

$$\rho(x) = \frac{1}{2}\|x - \Pi_K(x)\|_2^2, \quad x \in \mathbb{R}^n, \tag{1.19}$$

   is continuously differentiable in $\mathbb{R}^n$ with $\nabla\rho(x) = x - \Pi_K(x)$.

Note that by the first order necessary conditions for (1.15), we have

$$\langle \Pi_K(y) - y, x - \Pi_K(y) \rangle \geq 0, \quad \forall\, x \in K. \tag{1.20}$$

By (1.20) and the definition of $N(K, x)$, $x = \Pi_K(y)$ if and only if $y - x \in N(K, x)$. Hence it can be shown that $x^* \in$ SOL(K, F) if and only if it is a fixed point of the map $\Pi_K(I - \tau F)$ for any $\tau > 0$, that is,

$$x^* = \Pi_K(x^* - \tau F(x^*)). \tag{1.21}$$

(1.21) is true if and only if $-F(x^*) \in N_K(x^*)$.

In addition to the Euclidean norm, we sometimes use a vector norm induced by a symmetric positive definite matrix to define a skewed projection operator. Given a positive definite matrix $D \in \mathbb{R}^{n\times n}$, the D-norm of a vector $x \in \mathbb{R}^n$ is defined as

$$\|x\|_D = \sqrt{\langle x, Dx\rangle}. \tag{1.22}$$

Then the skewed projector $\Pi_{K,D}(x)$, of a vector x, on a closed convex subset K of $\mathbb{R}^n$ is the solution to

$$\begin{aligned} \min\ & \frac{1}{2}\langle y - x, D(y - x)\rangle, \\ & s.t. \quad y \in K. \end{aligned} \tag{1.23}$$

As in the case of euclidean norm, it follows from the first order conditions that x solves VI(K, F) if it is the fixed point of $\Pi_{K,D}(x - D^{-1}F(x))$ and vice versa.

1.2.1 Equation reformulations

Two nonsmooth equation reformulations of VI(K, F) can be defined with the aid of the following maps,

$$F_K^{nat}(x) = x - \Pi_K(x - F(x)), \tag{1.24}$$

and

$$F_K^{nor}(x) = F(\Pi_K(x)) + x - \Pi_K(x). \tag{1.25}$$

These maps are referred to as the natural and normal maps associated with VI(K, F) respectively. The following proposition is a combination of Propositions 1.5.8 and 1.5.9 of (Facchinei and Pang, 2003).

Proposition 1.1. *Let $K \subset \mathbb{R}^n$ be closed and convex and $F : \mathbb{R}^n \to \mathbb{R}^n$ be arbitrary. Then*

1. *$x \in$ SOL(K, F) $\Leftrightarrow F_K^{nat}(x) = 0$.*
2. *$x \in$ SOL(K, F) if and only if there exists z such that $x = \Pi_K(z)$ and $F_K^{nor}(z) = 0$.*

Another equation reformulation of VI(K, F) can be obtained from the Karush-Kuhn-Tucker

(KKT) conditions for the problem. In order to be able to write the KKT conditions for VI(K, F), we assume that K can be specified as

$$\begin{aligned} K = \{x \in \mathbb{R}^n : \quad & g(x) \leq 0, \\ & h(x) = 0\}, \end{aligned} \tag{1.26}$$

where $g : \mathbb{R}^n \to \mathbb{R}^m$ and $h : \mathbb{R}^n \to \mathbb{R}^l$ are continuously differentiable functions. Note that $x \in \mathbb{R}^n$ solves VI(K, F) if and only if x is an optimal solution to the problem

$$\min \langle F(x), y - x \rangle, \text{ subject to } y \in K.$$

Hence the KKT formulation for VI(K, F) is that for the aforementioned problem with x as the solution,

$$\begin{aligned} F(x) + \langle Jh(x)^T, \mu \rangle + \langle Jg(x)^T, \lambda \rangle &= 0, \\ h(x) &= 0, \\ 0 \geq g(x) \perp \lambda &\geq 0, \end{aligned} \tag{1.27}$$

where $\mu \in \mathbb{R}^l$, $\lambda \in \mathbb{R}^m$ are multipliers for the equality and inequality constraints respectively and $Jh(x)$, $Jg(x)$ are the Jacobian matrices for h and g respectively. To express (1.27) as a system of equations, we need to write the complementarity condition in an equality form. This can be achieved by using a C function.

Definition 1.1. A function $\psi : \mathbb{R}^2 \to \mathbb{R}$ is called a C function if for any $(a, b) \in \mathbb{R}^2$,

$$\psi(a, b) = 0 \Leftrightarrow [(a, b) \geq 0 \text{ and } ab = 0].$$

VI(K,F) can be written as

$$\phi(x,\mu,\lambda) = \begin{pmatrix} L(x,\mu,\lambda) \\ h(x) \\ C_F(-g_1(x),\lambda_1) \\ \vdots \\ C_F(-g_m(x),\lambda_m) \end{pmatrix} = 0, \tag{1.28}$$

where C_F is some C function, and $L(x,\mu,\lambda) = F(x) + \langle Jh(x)^T,\mu\rangle + \langle Jg(x)^T,\lambda\rangle$ is the Lagrangian function. An example of a C function is the $\min(\cdot,\cdot)$ function.

1.2.2 Merit functions

A merit function for a problem is a nonnegative function that takes a zero value only for the arguments that solve the problem. Formally, a merit function for VI(K, F) is defined as follows.

Definition 1.2. A merit function for VI(K, F) on a closed set $X \supseteq K$ is a nonnegative function $\theta : X \to \mathbb{R}_+$ such that $x \in \text{SOL}(K,F) \Leftrightarrow x \in X$ and $\theta(x) = 0$. In other words $x \in \text{SOL}(K,F)$ if and only if x solves

$$\min\ \theta(y), \quad y \in X \tag{1.29}$$

with the optimal objective value equal to 0.

A natural merit function, called the gap function, arises from the definition of VI(K, F). This function is defined on the domain D of F, and is given by

$$\theta_{gap}(x) = \sup_{y\in K}\langle F(x), x-y\rangle, \quad x \in D \supseteq K. \tag{1.30}$$

This is clearly a nonnegative extended value function on K. Since K is closed, finding $\theta_{gap}(x)$ is a concave maximization problem with a linear objective. Note that in general, this gap function is not differentiable, even in the simple case when the set K is polyhedral.

Some iterative procedures can be applied to minimize a merit function for finding its global minimum. But except in some cases, the merit functions are not convex, therefore an algorithm can

not guarantee to find the global minima. In general there must be some conditions on VI(K, F) in order for the stationary point of the gap function to be a solution to VI(K, F).

1.3 Solution analysis

We have already seen how VIs are related to optimization problems and the problem of solving a system of nonlinear equations. We also observed how a VI can be reformulated as a system of equations and/or a minimization problem. Results from analysis of such systems can then be applied to obtain the existence and uniqueness results of solutions to VI. In this section, we introduce several results on existence of the solution of a VI. These are well known results. We present them here and include their proofs for the sake of completeness. Interested reader is referred to (Facchinei and Pang, 2003)

Theorem 1.2. *If K is compact and convex and F is continuous on K, then VI(K, F) has at least one solution.*

Proof. According to Brouwer's fixed point theorem, for a continuous map $P : K \to K$, there is at least one $x^* \in K$ such that $P(x^*) = x^*$. Due to the continuity of F, $I - \tau F$ is continuous for $\tau > 0$. The projection operator $\Pi_K(x)$ is also continuous. Hence their composition $\Pi_K(x - F(x))$ is also continuous. The conclusion follows from compactness of K and the fact that a solution of $VI(\mathrm{K}, F)$ is a fixed point of the map $\Pi_K(x - F(x))$. □

If the feasible set K is not compact, the fixed point theorem is not applicable, but a solution to VI(K, F) exists under conditions given in the following theorem. Let $B_R(0)$ denote a closed ball of radius R centered at the origin and let $K_R = K \cap B_R(0)$.

Theorem 1.3. *VI(K, F) has a solution if and only if there exists $R > 0$ and a solution x_R^* of VI(K_R, F) with $\|x_R^*\| < R$.*

Proof. If $x \in \mathrm{SOL}(K, F)$, choose $R = \|x\| + 1$. Conversely, assume that x_R^* solves the $VI(\mathrm{K}_R, F)$ and $\|x_R^*\| < R$. If $x_R^* \notin \mathrm{SOL}(K, F)$, then there exists $y \in K$ such that

$$\langle F(x_R^*), y - x_R^* \rangle < 0. \tag{1.31}$$

Clearly, $\|y\| > R$. That is, the direction $y - x_R^*$ makes an obtuse angle with $F(x_R^*)$. Let $z = \beta(y-x)$ with $\beta > 0$ such that $\|x_R^* + z\| < R$. We can find such a $\beta > 0$ since $\|x_R^*\| < R$. Due to convexity of K, $x_R^* + z$ belongs to K and hence to K_R. We have

$\langle F(x_R^*), z - x_R^* \rangle = \langle F(x_R^*), z \rangle = \langle F(x_R^*), \beta(y-x) \rangle = \beta \langle F(x_R^*), (y-x) \rangle < 0.$

The last inequality follows from (1.31). This contradicts with the assumption that x_R^* solves VI(K_R, F). □

Corollary 1.4. *Suppose that F is coercive. That is, there exists $x_0 \in K$, such that*

$$\frac{\langle F(x) - F(x^0), x - x^0 \rangle}{\|x - x_0\|} \to \infty \tag{1.32}$$

as $\|x\| \to \infty$ for $x \in K$. Then VI(K, F) has a solution.

Uniqueness and existence results for VI(K, F) can be easily proven under various monotonicity assumptions. The following notions of monotonicity play important roles for analysing VIs. Let $K \subset \mathbb{R}^n$, and let $F : K \to \mathbb{R}^n$. Then F is said to be

(a) strongly monotone on K with constant $\tau > 0$ if for each pair of points $u, v \in K$, we have

$$\langle F(u) - F(v), u - v \rangle \geq \tau \|u - v\|^2;$$

(b) ξ monotone for some $\xi > 1$ on K with constant $\tau > 0$ if for each pair of points $u, v \in K$, we have

$$\langle F(u) - F(v), u - v \rangle \geq \tau \|u - v\|^\xi;$$

(c) strictly monotone on K if for each pair of points $u, v \in K$, $u \neq v$ we have

$$\langle F(u) - F(v), u - v \rangle > 0;$$

(d) monotone on K if for each pair of points $u, v \in K$, we have

$$\langle F(u) - F(v), u - v \rangle \geq 0;$$

(e) pseudomonotone on K if for each pair of points $u, v \in K$, we have

$$\langle F(v), u - v\rangle \geq 0 \Rightarrow \langle F(u), u - v\rangle \geq 0;$$

(f) quasimonotone on K if for each pair of points $u, v \in K$, we have

$$\langle F(v), u - v\rangle > 0 \Rightarrow \langle F(u), u - v\rangle \geq 0;$$

(g) explicitly quasimonotone on K if for all distinct $u, v \in K$, the following holds,

$$\langle F(v), u - v\rangle \geq 0 \Rightarrow \langle F(z), u - v\rangle > 0 \quad \text{for some } z \in (0.5(u + v), u).$$

It follows from the definitions that (a) $\Rightarrow$ (b) $\Rightarrow$ (c) $\Rightarrow$ (d) $\Rightarrow$(e) $\Rightarrow$ (f), and (g) $\Rightarrow$ (f).

The following results demonstrate that if F has certain monotonicity properties, the solution set to the VI possesses certain desirable characteristics.

Theorem 1.5. *Let $K \subset \mathbb{R}^n$ be closed and convex, and $F : K \to \mathbb{R}^n$ be continuous. Then the following hold.*

(a) If F is strictly monotone on K, then there is at most one solution to VI(K, F).

(b) If F is ξ monotone for some $\xi > 1$, then VI(K, F) has a unique solution.

The proof of (a) follows by the definition of VI and strict monotonicity. (b) follows from the fact that ξ monotonicity implies coercivity. Strict monotonicity does not guarantee existence of a solution to VI. Pseudo monotonicity of F gives the following result.

Theorem 1.6. *Let $K \subset \mathbb{R}^n$ be closed and convex, and $F : K \to \mathbb{R}^n$ be continuous. If F is pseudo monotone on K, then SOL(K, F) is convex.*

The proof follows from observing that pseudo monotonicity of F implies

$$\text{SOL}(K, F) = \bigcap_{y \in K} \{x \in K \mid \langle F(y), y - x\rangle \geq 0\}. \tag{1.33}$$

Since the set in the RHS of (1.33) is an intersection of convex sets, it is convex as well.

If in addition to being strongly monotne, $F(x)$ is also strongly Lipschitz continuous, then $\Pi_K(x - \tau F(x))$ is a contraction with respect to x and the operator $\Pi_K(x - \tau F(x))$ has a unique fixed point.

CHAPTER 2

Algorithms for variational inequalities

The algorithms for solving VI(K, F) can be classified into several categories depending upon which formulation a method exploits. There are methods based on KKT conditions, gap/merit functions, interior and smoothing methods, and projection based methods.

The algorithms for solving variational inequalities can also be categorized based on the sub-problems that are solved in each iteration. A general approach to solving VI(K, F) consists of creating a sequence $\{x^k\} \subset K$ such that each x^{k+1} solves VI(K, F^k),

$$\langle F^k(x^{k+1}), y - x^{k+1} \rangle \geq 0 \quad \forall\, y \in K, \tag{2.1}$$

where $F^k(\cdot)$ is some approximation to $F(x)$. F^k can be linear or nonlinear.

2.1 Linear approximation based methods

A linear F^k is of the form

$$F^k(x) = F(x^k) + \langle A(x^k), x - x^k \rangle. \tag{2.2}$$

As described by (Harker and Pang, 1990), different choices of $A(x^k)$ lead to different methods.

1. Newton's method: $A(x^k) = \nabla F(x^k)$.

2. Quasi Newton method: $A(x^k) \approx \nabla F(x^k)$.

3. Linearized Jacobi method: $A(x^k) = D(x^k)$, where $D(x^k)$ is the diagonal part of $\nabla F(x^k)$.

4. Successive over relaxation: $A(x^k) = T(x^k) + \frac{D(x^k)}{\omega^*}$, where $T(x^k)$ is the upper or lower triangular part of $\nabla F(x^k)$ and ω^* is a parameter in (0, 2).

5. Symmetrized Newton: $A(x^k) = \frac{1}{2}\{\nabla F(x^k) + \nabla F(x^k)^T\}$.

6. Projection method: $A(x^k) = G$, a symmetric positive definite matrix.

The convergence of these methods depends on x^* being a regular solution to VI(K, F).

Definition 2.1. (Robinson, 1980) Let x^* be a solution to VI(K, F), and F be differentiable at x^*. Then x^* is called a regular solution if there exists a neighborhood N of x^* and a scalar $\delta > 0$ such that for every y with $\|y\|_2 < \delta$, there is a unique vector $x(y) \in N$, Lispchitz continuous with respect to y, that solves the perturbed linearized VI(K, F^y) with $F^y : \mathbb{R}^n \to \mathbb{R}^n$ defined as

$$F^y(x) = F(x^*) + y + \langle \nabla F(x^*), x - x^* \rangle. \tag{2.3}$$

Let the set K be defined as in (1.26) with g_i, h_j being twice continuously differentaible for each i and j, and F being once continuously differentiable. Let $x^* \in$ SOL(K, F). Suppose that the following conditions hold.

1. There exist vectors $\mu^* \in \mathbb{R}^l$, and $\lambda^* \in \mathbb{R}^m$, such that $(x^*,\ \mu^*,\ \lambda^*)$ satisfy the KKT conditions for VI(K, F).

2. Linear independence constraint qualification (LICQ) holds at x^*. That is, the vectors $\{\nabla g_i(x^*) : i \in I_+ \cup I_0,\ \nabla h_j(x^*)\}$ are linearly independent, where $I_+ = \{i : \lambda_i^* > 0\}$ and $I_0 = \{i : g_i(x^*) = 0, \lambda_i^* = 0\}$.

3. The second order condition

$$\langle z, [\nabla F(x^*) + \sum_{i=1}^{l} \mu_i^* \nabla h_i^2(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i^2(x^*)] z \rangle > 0 \tag{2.4}$$

holds for all $z \neq 0$ such that,

$$\langle z, \nabla g_i(x^*) \rangle = 0 \qquad \forall i \in I_+,$$

$$\langle z, \nabla h_i(x^*) \rangle = 0 \qquad j = 1, 2, ..., l.$$

Then x^* is a regular solution to VI(K, F) (Robinson, 1980). If x^* is a regular solution to VI(K, F), then there exists a neighborhood N of x^* such that Newton's method converges to x^* as long as it starts from an initial point $x^0 \in N$ (Josephy, 1979b). Furthermore, if $\nabla F(x^*)$ is Lipschitz continuous around x^*, then the convergence rate is quadratic. But Newton's method for solving VI(K, F) suffers from the following drawbacks.

1. $\nabla F(x^*)$ needs to be evaluated at every step.

2. Each iteration requires solving a variational inequality subproblem.

3. The method converges only if the initial iterate is close enough to a solution.

Quasi Newton methods overcome the first drawback of Newton's method. For instance, secant methods (Josephy, 1979a) update the matrix $A(x^k)$ in each iteration by a simple small rank matrix. Although this reduces the work of finding $\nabla F(x^k)$ at each iteration, it does not make solving the subproblems any easier. Those methods can achieve a superlinear convergence rate at best.

Other linear approximation methods, including the linearized Jacobi method, symmetrized Newton method and projection algorithms, use a symmetric matrix $A(x_k)$ at each step. In these methods, the subproblem can be formulated as an optimization problem, thereby making it amenable to various optimization lagorithms. On the downside, those methods require stronger restrictions on the problem, and do not have quadratic rate of convergence. The linearized successive over relaxation method solves an LCP with a triangular matrix at each step. The following theorem summarizes the performance of the linearized Jacobi method and the symmetrized Newton's method.

Theorem 2.1. *(Chan and Pang, 1982) Let K be a nonempty, closed and convex subset of $\mathbb{R}^n$ and let F be a function from $\mathbb{R}^n$ to $\mathbb{R}^n$.*

1. *Suppose that F is once continuously differentiable, x^* solves VI(K, F) and $\nabla F(x^*)$ has positive diagonal elements. Let $D(x^*)$ and $B(x^*)$ be the diagonal and off diagonal parts of $\nabla F(x^*)$ respectively. If*

$$\|D(x^*)^{-1/2}B(x^*)D(x^*)^{-1/2}\|_2 < 1, \tag{2.5}$$

then there exists a neighborhood of x^* such that the sequence generated by linearized Jacobi method is well defined and converges to x^* if it starts with an initial point within that

neighborhood.

2. *Suppose that F is once continuously differentiable, x^* solves VI(K, F), and that $\nabla F(x^*)$ is positive definite. Let $A(x^*)$ and $C(x^*)$ be the symmetric and skew symmetric parts of $\nabla F(x^*)$ respectively. If*

$$\|C(x^*)\|_2 < \lambda_{min}(A(x^*)), \tag{2.6}$$

where $\lambda_{min}(A(x^*))$ denotes the least eigenvalue of $A(x^*)$, then there exists a neighborhood of x^* such that the sequence generated by the symmetrized Newton method is well defined and converges to x^* if it starts with an initial point within that neighborhood.

Moreover, the convergence rate of each of these methods is geometric, that is, there exists a constant $r \in (0, 1)$ such that for a certain vector norm and for all k,

$$\|x^{k+1} - x^*\| \leq r\|x^k - x^*\| \text{ holds.}$$

A class of methods applicable to VI(K, F) when K is a compact polyhedral set are the simplicial decomposition methods. Since K is a compact polyhedron, it can be expressed as the convex hull of its extreme points. At iteration k, a VI(K^k, F^k) is solved, where K^k denotes the convex hull of a subset of extreme points of K. A merit function is used to decide upon the addition or deletion of extreme points from K^k to obtain K^{k+1}. The effectiveness of the method depends on how many extreme points K has, and on the merit function used to guide the choice of extreme points at each iteration. (Lawphongpanich and Hearn, 1984) shows that if the gap function $\min_{y \in K} \langle F(x), y - x \rangle$ is used to control K^k, and $F^k = F$ for all iterations, then the method terminates in a finite number of major iterations if F is strongly monotone.

In what follows, we briefly describe methods based on the KKT formulations of variational inequalities, and proximal point methods.

2.2 KKT based methods

Methods based on the KKT formulation of VI(K, F) try to solve systems of nonsmooth constrained equations or minimize a merit function derived from (1.28). Using the $\min(\cdot, \cdot)$ and the

Fischer-Burmeister (FB) C-function,

$$\psi_{FB}(a,b) = \sqrt{a^2+b^2} - (a+b),\ \forall\ (a,b) \in \mathbb{R}^2,$$

we can obtain two different equation reformulations for VI(K, F). Let

$$\phi_{FB}(x,\mu,\lambda) = \begin{pmatrix} L(x,\mu,\lambda) \\ h(x) \\ C_{FB}(-g_1(x),\lambda_1) \\ \vdots \\ C_{FB}(-g_m(x),\lambda_m) \end{pmatrix},$$

$$\phi_{min}(x,\mu,\lambda) = \begin{pmatrix} L(x,\mu,\lambda) \\ h(x) \\ \min(-g_1(x),\lambda_1) \\ \vdots \\ \min(-g_m(x),\lambda_m) \end{pmatrix}.$$

A natural merit function for the KKT formulation is

$$\theta(x,\mu,\lambda) = \frac{1}{2}\langle \phi(x,\mu,\lambda), \phi(x,\mu,\lambda)\rangle,$$

where $\phi(x,\mu,\lambda)$ can be either $\phi_{FB}(x,\mu,\lambda)$ or $\phi_{\min}(x,\mu,\lambda)$. One can then try to solve the equation $\phi(x,\mu,\lambda) = 0$, or to minimize $\theta(x,\mu,\lambda)$. Algorithms based on the merit function $\theta(x,\mu,\lambda)$ can be regarded as special cases of interior point methods which use a more generic potential function $p(\phi(x,\mu,\lambda))$ to measure the improvement in each iteration. Methods based on the natural gap function (1.30) for the VI are a special case of Zhu and Marcotte's general framework for solving VIs, which will be described in a later section. The methods mentioned here find a descent direction for the merit or potential function at each iteration, and perform a line search routine in that direction to find the next iterate. One can refer to the book (Facchinei and Pang, 2003) for details on these algorithms.

2.3 Proximal point method

The proximal point method is another class of solution methods for VIs. This method solves VI(K, $F + \epsilon_k I$) at iteration k. Here $\{\epsilon_k\}$ is a sequence of positive scalars going to zero, and I is the identity map. If F is monotone, then $F + \epsilon I$ is strongly monotone. Thus each subproblem has a unique solution. (Rockafellar, 1976) showed that if ϵ_k are chosen according to an appropriate inexact rule, then the sequence $\{x^k\}$ is bounded if and only if SOL(K, F) $\neq \phi$. Moreover, if the sequence $\{x^k\}$ is bounded, then it converges to a solution of VI(K, F).

In the following generic proximal point scheme, VI(K, $F_{c,x}$), where $F_{c,x}(y) = y - x + cF(y)$ is solved inexactly at each iteration. The algorithm uses the fact that if F is monotone then the set valued map $F + N(K, \cdot)$ is maximal monotone. A set valued map $\phi : \mathbb{R}^n \to \mathbb{R}^n$ is (strongly) monotone if there exists a constant $c\,(>) \geq 0$ such that

$$\langle x - y, u - v \rangle \geq c\|x - y\|^2 \;\forall\; x, y \in dom(\phi) \text{ and } u \in \phi(x),\; v \in \phi(y). \tag{2.7}$$

A monotone map ϕ is maximal monotone if no monotone map ψ exists such that $graph\ \phi \subset graph\ \psi$. The following properties of maximal monotone maps ((Facchinei and Pang, 2003)) play an important role in the development of the algorithm being described here.

Theorem 2.2. *Let a set-valued map $\phi : \mathbb{R}^n \to \mathbb{R}^n$ be given. Then the following statements are equivalent.*

1. *ϕ is maximal monotone.*
2. *ϕ is maximal monotone and range$(I + \phi) = \mathbb{R}^n$.*
3. *For any positive c, $(I + c\phi)^{-1}$ is 1-co-coercive and dom$(I + c\phi)^{-1} = \mathbb{R}^n$.*
4. *For any positive c, $(I + c\phi)^{-1}$ is non-expansive and dom$(I + c\phi)^{-1} = \mathbb{R}^n$.*

The map $(I + c\phi)^{-1}$ is called the resolvent of ϕ, with constant c, and is denoted by $J_{c,\phi}$. It is single-valued and non expansive for a monotone ϕ, and its domain is $\mathbb{R}^n$ if ϕ is maximal monotone. The following Propositions suggest that solving VI(K, F) is equivalent to finding the fixed point of the resolvent of $T \equiv F + N(K, \cdot)$, provided T is maximal monotone.

Proposition 2.3. *For a maximal monotone ϕ, every positive c, and $x \in \mathbb{R}^n$, $0 \in \phi(x)$ if and only if $J_{c,\phi}(x) = x$.*

Proposition 2.4. *Let $K \subset \mathbb{R}^n$ be nonempty closed and convex and $F : K \to \mathbb{R}^n$ continuous. Then the following hold for $T \equiv F + N(K, \cdot)$.*

1. *$J_{c,T}(x)$ =SOL(K, $F_{c,x}$), where $F_{c,x} \equiv y - x + cF(y)$.*
2. *$N(K, \cdot)$ is maximal monotone.*
3. *If F is monotone then T is maximal monotone.*

If a maximal monotone map T has a zero, it can be obtained by using a fixed point recursion on the resolvent of T. Finding the resolvent is often a non-trivial problem itself. For instance, finding the resolvent to solve VI(K, F) is equivalent to solving VI(K, $I - x + F$). The following theorem states that finding the exact resolvent in the fixed point iteration is not necessary.

Theorem 2.5. *Let $T : \mathbb{R}^n \to \mathbb{R}^n$ be a maximal monotone map and let $x_0 \in \mathbb{R}^n$ be given. If T has a zero, the sequence defined by*

$$x^{k+1} = x^k + \rho_k(w^k - x^k), \tag{2.8}$$

where, for all k,

$$\|w^k - J_{c_k,T}(x^k)\| \leq \epsilon_k \tag{2.9}$$

and $\{\epsilon_k\} \subset [0, \infty]$ satisfies $\sum \epsilon_k < \infty$, $\rho_k \subset [R_m, R_M]$, where $0 < R_m \leq R_M < 2$, and $c_k \subset (C_m, \infty)$, where $C_m > 0$, converges to it.

Note that the subsequent iterates defined here are not the exact resolvents w^k of $J_{c_k T}$, but either an under relaxation ($\rho_k < 1$), or an over relaxation ($\rho_k > 1$), from the current iterate in the direction of w^k. The preceeding discussion leads to the following algorithm for VI(K, F) for a monotone F. Let $T \equiv F + K(N, \cdot)$.

Algorithm 2.1 Proximal point method for VIs

Initialization: Choose $x^0 \in \mathbb{R}^n$, $c_0 > 0$, sequences $\{\epsilon_k\}$, $\{\rho_k\}$, and $\{c_k\}$ as required by the previous theorem. Set $k = 1$, loop = 0.

while loop = 0 **do**

if $x_k \in$ SOL(K, F) **then**

Set loop $= 1$.

else

Find w_k such that $\|w^k - J_{c_k T}(x^k)\| \leq \epsilon_k$.

Set $x^{k+1} = x^k + \rho_k(w^k - x^k)$,

Set $k = k + 1$.

Select c_k, ϵ_k, and ρ_k.

end if

end while

If the VI has a solution, the algorithm converges to it. Otherwise the sequence generated by the algorithm is unbounded.

2.4 Projection based methods

We observed that x solves VI(K, F) if and only if

$$x = \Pi_{K,D}(x - D^{-1}F(x)) \tag{2.10}$$

where $\Pi_{K,D}$ is the skewed projector onto K defined by a $n \times n$ positive definite matrix D. If the projection map defined in (2.10) is a contraction, the sequence $\{x_k\}_{k=0}^{\infty}$ defined as $x_{k+1} = \Pi_{K,D}(x_k - D^{-1}F(x_k))$ converges to its fixed point irrespective of the choice of x_0.

Theorem 2.6. *((Facchinei and Pang, 2003)) Let K be a closed and convex subset of $\mathbb{R}^n$ and $F : K \to \mathbb{R}^n$ be μ monotone and Lipschitz continuous with constant L. If*

$$L^2\lambda_{max}(D) < 2\mu\lambda^2_{min}(D), \tag{2.11}$$

then the mapping $\Pi_{K,D}(x - D^{-1}F(x))$ *is a contraction from* K *to* K *with respect to the norm* $\|\cdot\|_D$*. Moreover, the sequence* $\{x^k\}$ *generated by the iterations*

$$x^{k+1} = \Pi_{K,D}(x^k - D^{-1}F(x^k)), \tag{2.12}$$

starting from any $x^0 \in K$*, converges to the solution of the VI(*K*,* F*) with a linear rate of convergence.*

One issue with the standard projection method is that it requires knowledge of constants characterizing Lipschitz continuity and strong monotonicity. The extragradient method requires a slightly weaker assumption on F, that is, F needs to be pseudomonotone. It requires two pojection calculations in each iteration:

$$\begin{aligned} x^{k+1/2} &= \Pi_K(x^k - \tau F(x^k)), \\ x^{k+1} &= \Pi_K(x^k - \tau F(x^{k+1/2})). \end{aligned}$$

The extragradient method still requires knowledge of the Lipschitz constant, L, of F, since it converges only if $\tau < L$. Due to the pseudo monotonicty of F, the hyperplane $\{x \in \mathbb{R}^n : \langle F(x^{k+1/2}), (x - x^{k+1/2})\rangle = 0\}$ separates x^k from any solution x^*. Consequently x^{k+1} is a move from x^k in a direction pointing towards SOL(K, F).

A projection based method by (He, 1997) uses the (2.10) formulation of a VI with $D = \beta I$, where β is a positive scalar. This algorithm requires F to be monotone. Define

$$\begin{aligned} e(u, \beta) &= u - \Pi_K[u - \beta F(u)], \\ d(u, \beta) &= e(u, \beta) - \beta[F(u) - F(\Pi_K(u - \beta F(u)))]. \end{aligned}$$

Algorithm 2.2 He's algorithm

Initialization: Choose $\gamma \in (0,2), \alpha, \delta \in (0,1)$ and $\beta > 0$. Choose an arbitrary $x^0 \in K$. Set $k = 1$, loop $= 0$.

while loop = 0 **do**

if $x_k \in$ SOL(K, F) **then**

Set loop $= 1$.

else

Set $\beta_k = \beta$.

while $e(x^k, \beta_k)^T d(x^k, \beta_k) < \delta \|e(x^k, \beta_k)\|^2$ **do**

$\beta_k := \alpha \beta_k$.

end while

Set $\beta = \beta_k$.

Set $g(x^k, \beta) = G^{-1} d(x^k, \beta)$.

Set $\rho(x^k, \beta) = \frac{e(x^k,\beta)^T d(x^k,\beta)}{\|g(x^k,\beta)\|^2}$.

Set $x^{k+1} = x^k - \gamma \rho(x^k, \beta) g(x^k, \beta)$

Set $k = k + 1$

end if

end while

Because the sequence $\{x^k\}$ generated by any contraction method is bounded and the mapping F is continuous, it is possible to prove that there is a $\beta_{min} > 0$ such that, for all $k, \beta_k \leq \beta_{min}$ and the method with Armijo's linesearch is well defined.

The hyperplane projection algorithm requires an even milder assumption on F, that of being pseudo monotone. The algorithm can be described as follows. Let $\tau > 0$ be a fixed scalar. Let $x^k \in K$ be given. First compute $\Pi_K(x^k - \tau F(x^k))$; call it y^k. Now search the line segment joining x^k and y^k for a point z^k such that the hyperplane

$$H^k = \{x \in \mathbb{R}^n \mid \langle F(z^k), x - z^k \rangle = 0\} \tag{2.13}$$

strictly separates x^k from SOL(K, F). Then project x^k onto H^k and the resulting point onto K, obtaining x^{k+1}. It can be shown that x^{k+1} is closer to SOL(K, F) than x^k. The algorithm is formalized as follows.

Algorithm 2.3 Hyperplane projection method

Initialization: Choose $x^0 \in K$, $\tau > 0$, and $\sigma \in (0, 1)$. Set $k = 1$. loop = 0.

while loop = 0 **do**

if $x_k \in$ SOL(K, F) **then**

Set loop $= 1$.

else

Set $y^k = \Pi_K(x^k - \tau F(x^k))$.

Find the smallest non negative integer i_k such that

$$\langle F(2^{-i}y^k + (1 - 2^{-i})x^k), x^k - y^k\rangle \geq \frac{\sigma}{\tau}\|x^k - y^k\|^2. \tag{2.14}$$

Set

$$z^k = 2^{-i_k}y^k + (1 - 2^{-i_k})x^k$$

Set

$$w^k = \Pi_{H^k}(x^k) = x^k - \frac{\langle F(z^k), x^k - z^k\rangle}{\|F(z^k)\|^2}F(z^k)$$

Set $x^{k+1} = \Pi_K(w^k)$.

$k = k + 1$

end if

end while

The common drawback with the methods described above is that they all require computation of projection on K, which is often not an easy task to perform. A method described by Fukushima (Fukushima, 1986) simplifies this task by taking projection on a half space containing K. He makes the following assumptions.

1. F is continuous on an open set containing K.

2. F is strongly monotone on an open set containing K.

3. For some $z \in K$, there exists a $\beta > 0$, and a bounded set $D \subset \mathbb{R}^n$ such that

$$\langle F(x), x - z \rangle \geq \beta \|F(x)\|_2 \quad \forall\, x \notin D. \tag{2.15}$$

4. The set $K = \{x \in \mathbb{R}^n \mid c(x) \leq 0\}$, where $c : \mathbb{R}^n \to \mathbb{R}$ is convex and the Slater condition is satisfied.

5. For any $x \in \mathbb{R}^n$, at least one subgradient $d \in \partial c(x)$ can be calculated, where $\partial c(x) = \{g \in \mathbb{R}^n \mid c(y) \geq c(x) + \langle g, y - x \rangle \quad \forall\, y \in \mathbb{R}^n\}$.

Fukushima's method is to take the projection of $x^k - \rho_k G^{-1} \frac{F(x^k)}{\|F(x^k)\|_2}$ on the halfspace T^k defined as

$$T^k = \{x \in \mathbb{R}^n \mid c(x^k) + \langle g^k, x - x^k \rangle \leq 0\} \tag{2.16}$$

where $\{\rho_k\}$ is a sequence of positive parameters satisfying $\lim_{k\to\infty} \rho_k = 0$ and $\sum_{k=1}^{\infty} \rho_k = \infty$. This algorithm does not require the starting point to be feasible. In fact, the intermediate solutions could also be infeasible in this method. The assumption (3.13), which is not satisfied by many VI problems, is crucial for convergence of this algorithm.

Another relaxed projection algorithm, given by (Cruz and Iusem, 2010) requires that F be paramonotone, that is, F be monotone and $\langle F(x) - F(y), x - y \rangle = 0$ implies $F(x) = F(y)$. This is a slightly weaker requirement than F being strongly monotone. This algorithm also requires (3.13) to be satisfied, and that the set K be represnted as $\{x \in \mathbb{R}^n \mid c(x) \leq 0\}$. The algorithm is defined as follows:

Algorithm 2.4 Relaxed projection method for paramonotone VIs

Initialization: choose $x^0 \in K$, and a sequence $\{\beta_k\}$, such that $\sum \beta_k = \infty$, and $\sum \beta_k^2 < \infty$. Set $k = 1$, loop = 0.

while loop = 0 **do**

if $x_k \in$ SOL(K, F) **then**

Set loop $= 1$.

else

Set $\eta_k = \max\{1, \|F(x^k)\|\}$.

Set $x^{k+1} = \Pi_{T^k}(x^k - \frac{\beta_k}{\eta_k} F(x^k))$.

Set $k = k + 1$

end if

end while

A generic gap function based framework given by (Zhu and Marcotte, 1994) encompasses several methods for solving monotone VIs. Let $\Omega(y, x) : K \times K \to \mathbb{R}$ be non-negative, continuously differentiable on $K \times K$, strongly convex on K with respect to y for all $x \in K$ and satisfy

$$\Omega(x, x) = 0 \text{ and } \nabla_y \Omega(x, x) = 0 \quad \forall\, x \in K. \tag{2.17}$$

Define

$$h(y, x) = \langle F(x), x - y \rangle - \Omega(y, x), \tag{2.18}$$

$$g(x) = \max_{y \in K} h(y, x) = h(H(x), x), \tag{2.19}$$

where $H(x)$ is the unique maximizer of $h(y, x)$ over K.

Since $h(x, x) = 0$, one always has $g(x) \geq 0$. If x solves VI(K, F), then $h(y, x) \leq 0$ implies $g(x) = 0$. Conversely, if $g(x) = 0$, then x solves $\max_{y \in K} h(y, x) = h(H(x), x)$. The variational

inequality formulation of the problem of finding $H(x)$ is

$$\begin{aligned} \langle \nabla_y h(x,x), z-y\rangle &\geq 0 \quad \forall\, z \in K \\ \text{i.e, } \langle F(x), z-y\rangle + \langle \nabla_y \Omega(x,x), z-y\rangle &\geq 0 \quad \forall\, z \in K \\ \text{i.e, } \langle F(x), z-y\rangle &\geq 0 \quad \forall\, z \in K, \end{aligned}$$

since $\nabla_y \Omega(x,x) = 0$. Hence $g(v)$ is a valid gap function for VI(K, F).

The problem of finding $H(x)$ is equivalent to solving the following auxiliary VI ($AVI(x)$)

$$\langle F(x) + \nabla_u \Omega(y,x), w-y\rangle \geq 0 \quad \forall\, w \in K. \tag{2.20}$$

Note that g is differentiable with gradient given by

$$\nabla g(x) = F(x) + \langle \nabla F(x), (x - H(x))\rangle - \nabla_y \Omega(H(x), x). \tag{2.21}$$

Under the assumptions that $F(x)$ is strongly monotone and Ω satisfies

$$\langle \nabla_y \Omega(y,x) + \nabla_x \Omega(y,x), y-x\rangle \geq 0 \quad \forall\, y, x \in K, \tag{2.22}$$

the following framework is proven to converge.

Algorithm 2.5 Zhu and Marcotte's generic framework

Initialization: choose $x^0 \in K$, and γ, β, σ in $(0, 1)$. Set $k = 1$, loop = 0.

while loop = 0 **do**

if $x_k \in$ SOL(K, F) **then**

Set loop $= 1$.

else

Solve $AVI(x^k)$ for $H(x^k)$.

Set $d^k = H(x^k) - x^k$.

if $g(x^k + d^k) < \gamma g(x^k)$ **then**

Set $x^{k+1} = x^k + d^k = H(x^k)$.

else

Select the smallest integer m such that

$$g(x^k) - g(x^k + \beta^m d^k) \geq -\sigma \beta^m \langle \nabla g(x^k), d^k \rangle,$$

Set $x^{k+1} = x^k + \beta^m d^k$.

Set $k = k + 1$

end if

end if

end while

If Ω is chosen carefully, then the AVI is easier to solve than the VI. The choice $\Omega = 0$ leads to an algorithm with the function (1.30) as the gap function. The choice $\Omega(y, x) = \langle y - x, G(y - x) \rangle$ corresponds to the regular projection method. The next chapter discusses a choice of Ω that makes the AVI amenable to relaxed projection methods.

CHAPTER 3

Two projection based algorithms

This chapter proposes two algorithms for solution of variational inequalities. Both of them belong to the broad category of projection based algorithms. The goal in developing these algorithms is twofold. First, the algorithms should converge under mild conditions, to be applicable to a general class of problems. Second, the subproblems in the iterations should be easy to solve. To this end, we present in Section 3.1 a modified relaxed projection based algorithm, which generates the halfspaces that are close to the boundary of the original set, using an anchor point. Halfspaces generated in such a way provide better approximations of the original set comparing to halfspaces generated by the iterate points, and therefore lead to fewer iterations in solution of variational inequalities. Section 3.2 presents a gap function based algorithm, which fits in the general framework introduced in (Zhu and Marcotte, 1994), with the special feature that the subproblems are solvable by relaxed projection methods. In section 3.3, we present an application of the descent framework to the oligopoly problem. The application leads to a special algorithm for box constrained variational inequalities, which requires taking projections on the box, and does not require knowledge of the monotonicity or Lipschitz constant of the function F.

3.1 An interior anchor point relaxed projection method

The original relaxed projection method proposed in (Fukushima, 1986), abbreviated as FRPM in the rest of this thesis, uses the iterate points to define halfspaces onto which the projections are made. (Censor and Gibali, 2008) proposed a modified method, by using an anchor point to define the halfspaces. The anchor point lies in the interior of the set K. This algorithm finds a support point between the current iterate point and the anchor point. This support point is located right on

the boundary of the set. Halfspaces generated by such support points provide better approximations of the original set. However, it is not possible to find the exact support points due to computational errors. Thus, in real implementation of such methods, one has to use an estimated support point. This section studies a method, which we call the interior anchor point relaxed projection method (IAPRPM), in which an estimated support point is found via a binary search. We prove that the algorithm still converges, and test the algorithm with numerical examples.

In order for the IAPRPM to converge, we make the following assumptions. Among these, K1, K2, F1 and F2 are assumed throughout this section. Assumption F3 will be used in proofs of some results.

Assumption K1. The defining set, K, is given as

$$K = \{x \in \mathbb{R}^n \mid c(x) \leq 0\}, \tag{3.1}$$

where $c : \mathbb{R}^n \to \mathbb{R}$ is a convex function.

Assumption K2. There exists a point $\bar{y}$ such that $c(\bar{y}) < 0$ (Slater condition).

Assumption F1. The function F is continuous on an open set containing K.

Assumption F2. The function F is strongly monotone on an open set containing K.

Assumption F3. There exist a $z \in K$, $\beta > 0$, and a bounded set $D \subset \mathbb{R}^n$ such that

$$\langle F(x), x - z\rangle \geq \beta \|F(x)\|_2 \quad \forall\, x \notin D. \tag{3.2}$$

Note that if K is specified as

$$K = \{x \in \mathbb{R}^n \mid c_i(x) \leq 0 \; i = 1, \ldots, r\}, \tag{3.3}$$

we can define c as $c(x) = \min\limits_{i=1,\ldots,r} c_i(x)$.

For each $x \in \mathbb{R}^n$, we define

$$H(x) = \{y \in \mathbb{R}^n \mid c(x) + \langle g, y - x\rangle = 0\}, \tag{3.4}$$

where $g \in \delta c(x)$, the set of subgradients of $c(x)$. The set $\delta c(x)$ is defined as

$$\delta c(x) = \{g \in \mathbb{R}^n \mid c(y) \geq c(x) + \langle g, y - x \rangle, \ \forall y \in \mathbb{R}^n\}. \tag{3.5}$$

Note that, strictly speaking, the above $H(x)$ should be replaced as $H(x, g)$, as it depends not only on x but also on the arbitrarily chosen subgradient g. However, for notational convenience we write it as $H(x)$, and adopt the convention that for each x a single subgradient is selected. In what follows, we present some preliminary results that stem from convexity of K, followed by a description of the IAPRPM algorithm and the proof of its covegrence.

Lemma 3.1. *Let $x \in \mathbb{R}^n$, and define the half space*

$$H^-(x) = \{y \in \mathbb{R}^n \mid c(x) + \langle g, y - x \rangle \leq 0\},$$

where g is a subgradient of K at x. Then $K \subset H^-(x)$.

Proof. Let $y \in K$. Then $c(y) \leq 0$. By the definition of subgradient, we have

$$c(x) + \langle g, y - x \rangle \leq c(y).$$

Hence $y \in H^-(x)$. □

Lemma 3.2. *Let $x \in \mathbb{R}^n$ with $c(x) > 0$. Then the hyperplane $H(x)$ strictly separates x from K.*

Proof. Each $s \in K$ satisfies

$$c(x) + \langle g, s - y \rangle \leq c(s) \leq 0.$$

On the other hand

$$c(x) + \langle g, x - x \rangle = c(x) > 0.$$

Hence the hyperplane $H(x)$ strictly separates x from K. □

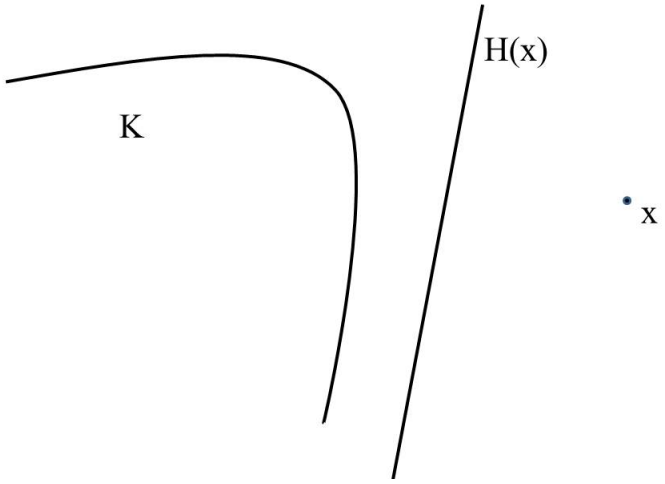

Figure 3.1: $H(x)$ strictly separates x from K

Proposition 3.3. *Let $x \notin K$, and let $\bar{y}$ be the point in assumption K2. Let L be the line segment joining x and $\bar{y}$. If $z \in L \setminus K$, then $H^-(z)$ contains K, and $H(z)$ separates x from K strictly.*

Proof. That $H^-(z)$ contains K follows directly from Lemma 3.1.

By Lemma 3.2, $z \notin H^-(z)$. Since $z \in L \setminus K$, we can assume that z is a convex combination of x and $\bar{y}$. That is,

$$z = \lambda x + (1 - \lambda)\bar{y}$$

for some $\lambda \in (0, 1)$. If $x \in H^-(z)$, then we have

$$c(z) + \langle g, x - z \rangle \leq 0$$

and

$$c(z) + \langle g, \bar{y} - z \rangle \leq 0$$

since $\bar{y} \in K \subset H^-(z)$. Multiplying the first inequality with λ, second with $1-\lambda$ and adding, we get

$$
\begin{aligned}
& \lambda(c(z) + \langle g, x - z\rangle) + (1-\lambda)(c(z) + \langle g, \bar{y} - z\rangle) \\
= \;& c(z) + \langle g, \lambda x + (1-\lambda)\bar{y} - z\rangle \\
= \;& c(z) + \langle g, z - z\rangle \\
= \;& c(z) \leq 0
\end{aligned}
$$

which is a contradiction since $c(z) > 0$. □

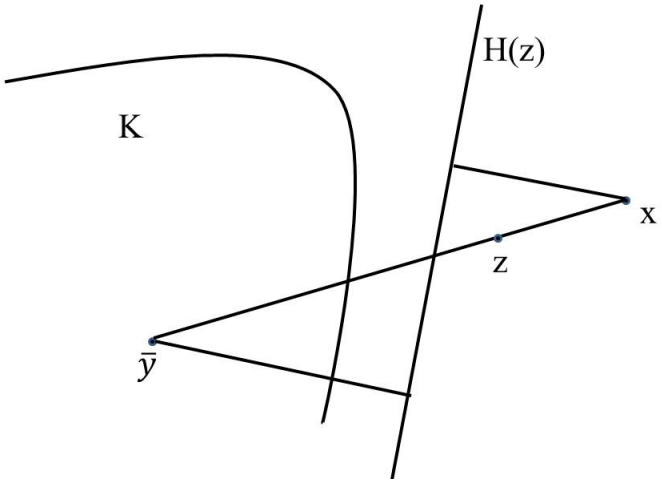

Figure 3.2: $H(z)$ separates x from K

The previous Lemmas establish important properties of the halfspace that approximates K for the following algorithm. The algorithm performs different computations depending upon feasibility of the current iterate.

Algorithm 3.1 Interior anchor point RPM

Initialization: Select $x^0 \in \mathbb{R}^n$, a small $\epsilon > 0$, a sequence $\{\rho_k\}$ with $\rho_k \to 0$, and $\sum_{k=1}^{\infty} \rho_k = \infty$ and a positive definite matrix G. Set $k = 1$, loop $= 0$.

while loop = 0 **do**

Find the point to be projected as

$$p^k = \begin{cases} x^k - \rho_k G^{-1} \frac{F(x^k)}{\|F(x^k)\|_2} & \text{if } F(x^k) \neq 0 \\ x^k & \text{if } F(x^k) = 0. \end{cases}$$

if $x^k \in K$ **then**

set $x^{k+1} = \Pi_{H^-(x^k),G}(p^k)$.

else

Construct the line segment L_k from $\bar{y}$ to x^k. Find a point w^k on L_k with $c(w^k) > 0$ and $dist(w^k, K) \leq \epsilon_k$ via a binary serach between x^k and $\bar{y}$.

Set $x^{k+1} = \Pi_{H^-(w^k),G}(p^k)$.

end if

if $x^{k+1} = x^k$ **then**

set loop $= 0$.

end if

set $k = k + 1$.

end while

Algorithm 3.1 involves finding the skewed projection of a point on a halfspace S, which has a closed form formula. Let a halfspace S be of the form,

$$S = \{x \in \mathbb{R}^n | \langle g, x \rangle \leq c\}. \tag{3.6}$$

For a point $p \in \mathbb{R}^n$ and a positive definite matrix G, the KKT conditions for the problem of finding

$\Pi_{S,G}(p)$ are

$$\begin{aligned} G(y-p)+\mu g &= 0, \\ \langle g, y\rangle &\leq c, \\ \mu \times (\langle g, x\rangle - c) &= 0, \\ \mu &\geq 0. \end{aligned} \tag{3.7}$$

If $p \in S$, $\Pi_{S,G}(p) = p$. Otherwise $\Pi_{S,G}(p)$ is on the boundary of S, that is

$$\langle g, \Pi_{S,G}(p)\rangle = c. \tag{3.8}$$

From (3.7), we get

$$\Pi_{S,G}(p) = -\mu G^{-1} g + p.$$

From the above equation and (3.8), we get

$$\begin{aligned} & \langle g, -\mu G^{-1} g + p\rangle = c \\ \text{i.e.} \quad & \mu = \tfrac{\langle g,p\rangle - c}{\|g\|_G^2} \\ \text{i.e.} \quad & \Pi_{S,G}(p) = -G^{-1} g \tfrac{\langle g,p\rangle - c}{\|g\|_G^2} + p \end{aligned}$$

That is,

$$\Pi_{S,G}(p) = \min\{0, \frac{c - \langle g, p\rangle}{\|g\|_G^2}\} \times G^{-1} g + p.$$

The proof of convergence of Algorithm 3.1 follows from the following Lemmas and Theorems. The proof here follows along the path in (Censor and Gibali, 2008). However, we made some major changes to handle the inexactness of the anchor point. The proof can be broken down in three major steps.

Step 1. Prove that the sequence of iterates produced by Algorithm 3.1 is bounded. The assumption F3 plays an important role here.

Step 2. Prove that the iterates get closer to the set K, and closer to each successive iterate. Lemmas 3.11, 3.12 and Proposition 3.13 lead to those results. Lemma 3.11 shows a property of the

binary search routine which leads to Lemma 3.12 and Proposition 3.13 which compare the distance of iterates from the half space with that from the set K.

Step 3. Prove that the sequence converges to a solution.

Lemma 3.4. *If $x^{k+1} = x^k$, then $x^k \in K$.*

Proof. If $x^k \notin K$, then due to Proposition (3.3), $x^k \notin H^-(w^k)$. But

$$x^{k+1} = \Pi_{H^-(w^k),G}(p^k) \in H^-(w^k).$$

Hence $x^{k+1} \neq x^k$. □

Lemma 3.5. *If $x^{k+1} = x^k$, then $x^k \in SOL(K, F)$.*

Proof. From Lemma 3.4, $x^k \in K$. If $x^k \in int(K)$, then $x^{k+1} = x^k$ if and only if $F(x^k) = 0$. Hence x^k solves VI(K, F).

If x^k is on the boundary of K and $F(x^k) = 0$, then x^k solves VI(K, F). Otherwise, we have $\Pi_{H^-(x^k),G}(p^k) = x^k$ where

$$p^k = x^k - \rho_k G^{-1} \frac{F(x^k)}{\|F(x^k)\|_2}.$$

That is x^k is the solution to VI$(H^-(x^k), F)$. Hence $-F(x^k) \in N_{H(x^k)}(x^k)$. Since $K \subset H^-(x^k)$, $N_{H^-(x^k)}(x^k) \subset N_K(x^k)$. Therefore $-F(x^k) \in N_K(x^k)$. That is, x^k solves VI(K, F). □

Lemma 3.6. *Let $y \in \mathbb{R}^n$ be an arbitrary point. Then in Algorithm 3.1, for all $k \geq 1$, the following holds*

$$\|\Pi_{H^-(w^k),G}(y) - x\|_G^2 \leq \|y - x\|_G^2 - \|\Pi_{H^-(w^k),G}(y) - y\|_G^2 \; \forall x \in K. \tag{3.9}$$

Proof. The proof follows from Lemma 10 of (Censor and Gibali, 2008) with $A = K$, $E = H^-(w^k)$ and $P = \mathbb{R}^n$. □

For completeness, we state Lemma 10 of (Censor and Gibali, 2008) here.

Lemma 3.7. *Let A, E and P be nonempty closed convex sets in $\mathbb{R}^n$, such that $A \subset E \subset P$ and G be a positive definite matrix. For any $x \in F$, let y be the point in E closest to x. Then we have,*

$$\|y - z\|_G^2 \leq \|x - z\|_G^2 - \|y - x\|_G^2 \; \forall z \in A. \tag{3.10}$$

Furthurmore,

$$(dist_G(y, A))^2 \leq (dist_G(x, A))^2 - \|y - x\|_G^2. \tag{3.11}$$

The next lemma is quoted from (Fukushima, 1986).

Lemma 3.8. *Let $\{a_k\}$ and $\{b_k\}$ be sequences of nonnegative numbers and let $\mu \in (0, 1)$ be a constant. If the inequalities*

$$a_{k+1} \leq \mu a_k + b_k \ \forall k \geq 0 \tag{3.12}$$

hold and if $b_k \to 0$, then $a_k \to 0$.

Lemma 3.9. *If assumption F3 holds, that is, there exists a $z \in K$, $\beta > 0$, and a bounded set $D \subset \mathbb{R}^n$ such that*

$$\langle F(x), x - z\rangle \geq \beta \|F(x)\|_2 \quad \forall\, x \notin D, \tag{3.13}$$

then the sequence of iterates $\{x^k\}$ generated by Algorithm 3.1 is bounded.

Proof. This proof uses similar arguments as in Lemma 13 of (Censor and Gibali, 2008). Let z be the point in assumption F3. From (3.10) of Lemma 3.7, we have

$$\|\Pi_{H^-(w^k),G}(p^k) - z\|_G^2 \leq \|p^k - z\|_G^2. \tag{3.14}$$

Consider the case when $x^k \notin D$. If $F(x^k) = 0$, then

$$x^{k+1} = \Pi_{H^-(w^k),G}(p^k) = \Pi_{H^-(w^k),G}(x^k),$$

that is

$$\| x^{k+1} - z\|_G^2 \leq \|x^k - z\|_G^2. \tag{3.15}$$

Otherwise, from non expansiveness of projector operator

$$\begin{aligned} \| x^{k+1} - z\|_G^2 &\leq \|x^k - \rho_k G^{-1} \frac{F(x^k)}{\|F(x^k)\|_2} - z\|_G^2 \\ &= \|x^k - z\|_G^2 - 2\frac{\rho_k}{\|F(x^k)\|_2}\langle F(x^k), x^k - y\rangle \\ &\quad + \frac{\rho_k^2}{\|F(x^k)\|_2^2}\langle F(x^k), G^{-1}F(x^k)\rangle \\ &= \|x^k - y\|_G^2 - 2\rho_k\beta + \rho_k^2\nu^{-1} \\ &= \|x^k - y\|_G^2 - \rho_k(2\beta - \rho_k\nu^{-1}) \end{aligned} \tag{3.16}$$

where ν is the smallest eigenvalue of G. since $\rho_k \to 0$, for large enough k, we have

$$\| x^{k+1} - z\|_G^2 \leq \|x^k - z\|_G^2 \tag{3.17}$$

For the case when $x^k \in D$, we can apply triangle inequality to the first line of (3.16) to get

$$\begin{aligned} \| x^{k+1} - z\|_G &\leq \|x^k - \rho_k G^{-1} \frac{F(x^k)}{\|F(x^k)\|_2} - z\|_G \\ &\leq \|x^k - z\|_G + \frac{\rho_k\sqrt{\langle G^{-1}F(x^k), F(x^k)\rangle}}{\|F(x^k)\|_2} \\ &\leq \|x^k - z\|_G + \rho_k\nu^{0.5} \\ &= \|x^k - y\|_G + \epsilon \end{aligned} \tag{3.18}$$

for sufficiently large k. Here, $\epsilon > 0$ is a small constant. Hence the sequence generated is bounded.

□

Lemma 3.10. *Let x^k be an iterate for Algorithm 3.1. Let L^k be the line segment connecting x^k and $\bar{y}$, and i^k be the point of intersection of K and L^k. If we use binary search on L^k to find a point w^k on it such that $w^k \notin K$, then we have*

$$\frac{\|x^k - w^k\|_2}{\|x^k - i^k\|_2} \geq \frac{1}{2}. \tag{3.19}$$

Proof. The following figure illustrates the situation.

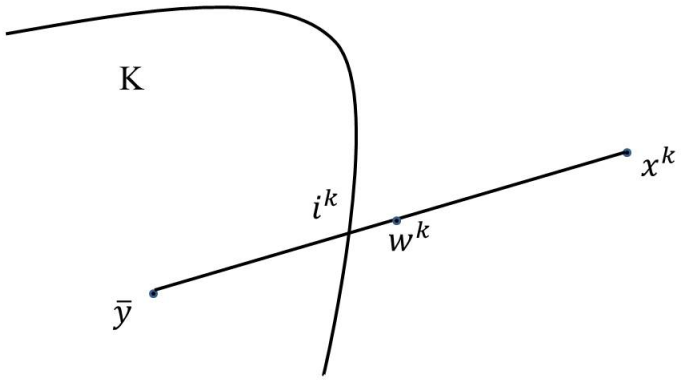

Figure 3.3: w^k is closer to i^k than x^k

The binary search will generate a finite sequence $u^1, u^2, ..., u^r (= w^k)$ with $w^k = \frac{u^{r-1}+x^k}{2}$. Since the search terminates at u^r, $u^{r-1} \in K$. That is,

$$\|x^k - u^{r-1}\|_2 \geq \|x^k - i^k\|_2.$$

Therefore $\|x^k - w^k\|_2 \geq \|x^k - w'\|_2$, where $w' = \frac{(x^k+i^k)}{2}$ is the mid point of the line segment joining x^k and i^k. That is,

$$\frac{\|x^k - w^k\|_2}{\|x^k - i^k\|_2} \geq \frac{\|x^k - w'\|_2}{\|x - i^k\|_2} \geq \frac{1}{2}.$$

□

Lemma 3.11. *Let $\bar{y}$ be the point satisfying Slater condition, that is $c(\bar{y}) < 0$. Also, let G be a given positive definite matrix, and $\{x^k\}$ be a sequence generated by Algorithm 3.1. Then there exists a $\delta \in (0, 1]$ such that*

$$dist_G(x^k, K) \times \delta \leq dist_G(x^k, H^-(w^k)) \; \forall k. \tag{3.20}$$

Proof. If $x^k \in K$, both sides of the given inequality are zero. We assume that $x^k \notin K$. We will first prove the result for $G = I$.

Let a^k be the projection of x^k on $H(w^k)$, b^k be that of $\bar{y}$ on $H(w^k)$, c^k be the point of intersection of the line connecting x and $\bar{y}$, and $H(w^k)$, and i^k be the point of intersection of the line

connecting x^k and $\bar{y}$, and K. The following figure illustrates the situation.

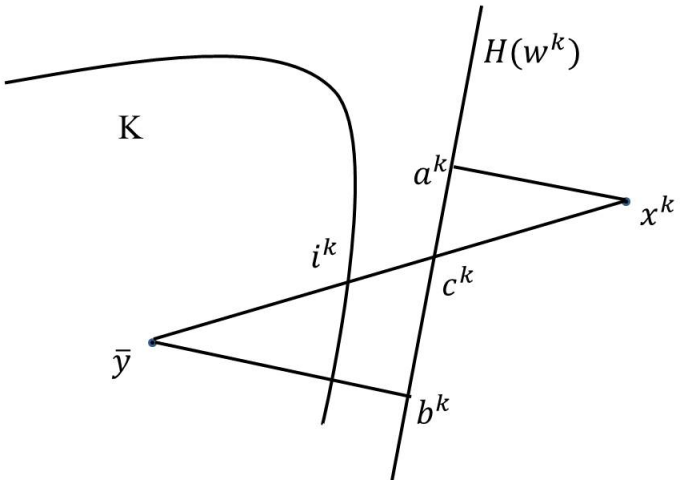

Figure 3.4: $dist(x^k, K)$ is related to that $dist(x^k, H^-(w^k))$

We will drop the super-scripts for the sake of brevity. Since the triangles $b\bar{y}c$ and axc are similar, we have

$$\|a - x\|_2 = \frac{\|x - c\|_2 \|\bar{y} - b\|_2}{\|\bar{y} - c\|_2}. \tag{3.21}$$

Let $d = dist(\bar{y}, bd(K))$. Clearly $\|\bar{y} - b\|_2 \geq d > 0$. Note that since the sequence $\{x^k\}$ is bounded (Lemma 3.9), so is the sequence $\{\bar{y} - c^k\}$. So there exists an $N > 0$ such that $\|\bar{y} - c^k\|_2 \leq N$ for all k. Hence,

$$\|a - x\|_2 \geq \frac{d}{N} \|x - c\|_2. \tag{3.22}$$

Since $x \notin K$, the hyperplane $H^-(x)$ separates x from K strictly. Hence

$$\|x - c\|_2 \geq \|x - w\|_2.$$

From Lemma 3.10, we have

$$\|x - w\|_2 \geq \frac{1}{2} \|x - i\|_2.$$

Hence

$$\|x - c\|_2 \geq \frac{1}{2} \|x - i\|_2.$$

Also $\|x - i\|_2 \geq dist(x, K)$. These inequalities, together with (3.22), imply

$$dist(x^k, H^-(w^k)) = \|a - x\|_2 \geq \frac{d}{2N} dist(x, K), \tag{3.23}$$

which is the same as (3.28) with $\delta = \frac{d}{2N}$. N can be made larger to make $\delta < 1$.

For the general case, when $G \neq I$, we use the equivalence of norms and (3.23) for the proof. It is known that

$$M_1 \|x\|_G \leq \|x\|_2 \leq M_2 \|x\|_G, \tag{3.24}$$

for some constants M_1 and M_2. So we have

$$dist_G(x, H^-(w)) = \|x - \Pi_{H^-(w),G}(x)\|_G \geq \frac{\|x - \Pi_{H^-(w),G}(x)\|_2}{M_2}.$$

Since

$$\|x - \Pi_{H^-(w),G}(x)\|_2 \geq \|x - \Pi_{H^-(w)}(x)\|_2 = dist(x, H^-(w)),$$

we have

$$dist_G(x, H^-(w)) \geq \frac{dist(x, H^-(w))}{M_2}. \tag{3.25}$$

Using a similar argument, we observe that

$$dist(x, K) = \|x - \Pi_K(x)\|_2 \geq M_1 \|x - \Pi_K(x)\|_G \geq M_1 \|x - \Pi_{K,G}(x)\|_G. \tag{3.26}$$

From (3.23), (3.25), and (3.26), we get

$$dist_G(x, H^-(w)) \geq \frac{d}{2N} \frac{M_1}{M_2} dist_G(x, K), \tag{3.27}$$

which is the same as (3.28) with $\delta = \frac{dM_1}{2NM_2}$. □

Proposition 3.12. *Assume that K satisfies the Slater condition. Also, let G be a given positive definite matrix, and $\{x^k\}$ be a sequence generated by Algorithm 3.1. Then there exists a $\mu \in [0, 1)$ such that*

$$dist_G(\Pi_{H^-(w^k),G}(x^k), K) \leq \mu \times dist_G(x^k, K) \ \forall k. \tag{3.28}$$

Proof. Let $u^k = \Pi_{H^-(w^k),G}(x^k)$. By (3.11) of Lemma 3.7

$$dist_G(u^k, K)^2 \leq dist_G(x^k, K)^2 - \|u^k - x^k\|_G^2. \tag{3.29}$$

From Lemma 3.11, we have a $\delta > 0$ such that

$$dist_G(x^k, K) \times \delta \leq dist_G(x^k, H^-(w^k)) \ \forall k. \tag{3.30}$$

Since $0 < \delta \leq 1$, we have

$$dist_G(x^k, K)^2 \times -\delta^2 \geq -dist_G(x^k, H^-(w^k))^2 \ \ \forall k. \tag{3.31}$$

Note that $dist_G(x^k, H^-(w^k)) = \|u^k - x^k\|_G$. From (3.31) and (3.29), we get

$$\begin{aligned} dist_G(u^k, K)^2 &\leq dist_G(x^k, K)^2 + dist_G(x^k, K)^2 \times -\delta^2 \\ &= (1 - \delta^2) dist_G(x^k, K)^2 \end{aligned} \tag{3.32}$$

which is the same as (3.28) with $\mu = \sqrt{1 - \delta^2}$. □

Lemma 3.13. *For any sequence $\{x^k\}$ generated by Algorithm 3.1, we have*

$$\lim_{k \to \infty} dist(x^k, K) = 0.$$

Proof. The arguments in this proof are similar to the proof of Lemma 15 of (Censor and Gibali, 2008).

If $F(x^k) = 0$, then $x^{k+1} = \Pi_{H^-(w^k),G}(x^k)$. Hence from Proposition 3.12, we have

$$dist_G(x^{k+1}, K) \leq \mu \times dist_G(x^k, K). \tag{3.33}$$

Otherwise, from (3.29), we get

$$dist_G(\Pi_{H^-(w^k),G}(x^k), K)^2 \leq dist_G(x^k, K)^2 - \|\Pi_{H^-(w^k),G}(x^k) - x^k\|_G^2 \ \forall k. \tag{3.34}$$

Also, there exists a $\mu \in [0, 1)$ such that

$$dist_G(\Pi_{H^(w^k),G}(x^k), K) \leq \mu \times dist_G(x^k, K). \tag{3.35}$$

From the non expansiveness of projector, we get

$$\begin{aligned}
\|x^{k+1} - \Pi_{H^-(w^k)}(x^k)\|_G^2 &= \|\Pi_{H^-(w^k),G}(p^k) - \Pi_{H^-(w^k)}(x^k)\|_G^2 \\
&\leq \|p^k - x^k\|_G^2 \\
&\leq \|x^k - \rho_k G^{-1} \frac{F(x^k)}{\|F(x^k)\|_2} - x^k\|_G^2 \\
&\leq \|\rho_k G^{-1} \frac{F(x^k)}{\|F(x^k)\|_2}\|_G^2 \\
&\leq \rho_k^2 \nu^{-1},
\end{aligned}$$

where ν is the smallest eigenvalue of G. That is

$$\|x^{k+1} - \Pi_{H^-(w^k)}(x^k)\|_G \leq \frac{\rho_k}{\sqrt{\nu}}. \tag{3.36}$$

By triangle inequality with G norms, we get

$$\begin{aligned}
&\|x^{k+1} - \Pi_{K,G}(\Pi_{H^-(w^k)}(x^k))\|_G \\
&= \|x^{k+1} - \Pi_{H^-(w^k)}(x^k) + \Pi_{H^-(w^k)}(x^k) - \Pi_{K,G}(\Pi_{H^-(w^k)}(x^k))\|_G \\
&\leq \|x^{k+1} - \Pi_{H^-(w^k)}(x^k)\|_G + \|\Pi_{H^-(w^k)}(x^k) - \Pi_{K,G}(\Pi_{H^-(w^k)}(x^k))\|_G.
\end{aligned}$$

Since $\Pi_{K,G}(\Pi_{H^-(w^k)}(x^k)) \in K$,

$$dist_G(x^{k+1}, K) \leq \|x^{k+1} - \Pi_{K,G}(\Pi_{H^-(w^k)}(x^k))\|_G. \tag{3.37}$$

From the last three inequalities, we get

$$\begin{aligned}
&\text{dist}_G(x^{k+1}, K) \\
&\leq \|x^{k+1} - \Pi_{H^-(w^k)}(x^k)\| + dist_G(\Pi_{H^-(w^k)}(x^k), K) \\
&\leq \tfrac{\rho_k}{\sqrt{\nu}} + \mu \times dist(x^k, K),
\end{aligned}$$

where the last inequality follows from Proposition 3.12. The result follows from Lemma 3.8.

□

Lemma 3.14. *For any sequence $\{x^k\}$ generated by Algorithm 3.1, we have*

$$\lim_{k\to} \|x^{k+1} - x^k\|_G = 0.$$

Proof. The arguments in this proof is similar to the proof of Lemma 16 of (Censor and Gibali, 2008).

If $F(x^k) = 0$,

$$\begin{aligned} \|x^{k+1} - x^k\|_G &= \|\Pi_{H^-(w^k),G}(x^k) - x^k\|_G \\ &= dist_G(x^k, H^-(w^k)) \\ &\leq dist_G(x^k, K). \end{aligned} \tag{3.38}$$

If there is a subsequence $\{x^r\}$, with $r \in R \subset \{1, 2, ...\}$, of $\{x_k\}$ with $F(x^r) = 0$, then from the above inequality and the previous Lemma, we get $\|x^{r+1} - x^r\|_G \to 0$.

If $F(x^k) \neq 0$,

$$\begin{aligned} \|x^{k+1} - x^k\|_G &= \|x^{k+1} - \Pi_{H^-(w^k),G}(x^k) + \Pi_{H^-(w^k),G}(x^k) - x^k\|_G \\ &= \|x^{k+1} - \Pi_{H^-(w^k),G}(x^k)\|_G + \|\Pi_{H^-(w^k),G}(x^k) - x^k\|_G \\ &\leq \frac{\rho_k}{\sqrt{\nu}} + dist_G(x^k, H^-(w^k)) \\ &\leq \frac{\rho_k}{\sqrt{\nu}} + dist_G(x^k, K) \end{aligned} \tag{3.39}$$

where the last inequality follows from the fact that $H^-(w^k) \subset K$. Since both $dist_G(x^k, K)$ and ρ_k go to zero, the proof is complete. □

Theorem 3.15. *The sequence generated by Algorithm 3.1 converges to a solution x^* of VI(K, F).*

Proof. The proof is same as that in (Censor and Gibali, 2008) and (Fukushima, 1986). □

The comparison of the performances of Fukushima's relaxed projection method versus IAPRPM for some examples are given below. The results are given in table 3.1. Here n is the number of

variables, m is the number of constraints, and 1^n refers to the vector in $\mathbb{R}^n$ with all components equal to 1.

Example 3.1. $n = 3, m = 1$,

$F(x) = x - 1$,

$K = \{x \in \mathbb{R}^3 | \sum_{i=1}^{3} x_i^2 \leq 1\}$,

$x_0 = 0 \times 1^n$,

Solution $= 0.5774 \times 1^n$.

Example 3.2. $n = 5, m = 1$,

$F(x) = x - 1$,

$K = \{x \in \mathbb{R}^5 | \sum_{i=1}^{5} x_i^2 \leq 2\}$,

$x_0 = 0 \times 1^n$,

Solution $= 0.6325 \times 1^n$.

Example 3.3. $n = 10, m = 1$,

$F(x) = x - 1$,

$K = \{x \in \mathbb{R}^{10} | \sum_{i=1}^{10} x_i^2 \leq 5\}$,

$x_0 = 0 \times 1^n$,

Solution $= 0.7071 \times 1^n$.

Example 3.4. $n = 20, m = 1$,

$F(x) = x - 1$,

$K = \{x \in \mathbb{R}^{20} | \sum_{i=1}^{20} x_i^2 \leq 15\}$,

$x_0 = 0 \times 1^n$,

Solution $= 0.8660 \times 1^n$.

Example 3.5. $n = 3, m = 1$,

$F_i(x) = e^{x_i}$,

$K = \{x \in \mathbb{R}^3 | \sum_{i=1}^{3} x_i^2 \leq 1\}$,

$x_0 = 0 \times 1^n$,

Solution $= -0.5774 \times 1^n$.

Example 3.6. $n = 3, m = 1$,

$F_i(x) = x_i^3 - 1$,

$K = \{x \in \mathbb{R}^3 | \sum_{i=1}^{3} x_i^2 \leq 1\}$,

$x_0 = 0 \times 1^n$,

Solution $= 0.8660 \times 1^n$.

Example 3.7. $n = 3, m = 1$,

$$F(x) = \begin{pmatrix} x_1 + 0.2x_1^3 - 0.5x_2 + 0.1x_3 - 4 \\ -0.5x_1 + x_2 + 0.1x_2^3 + 0.5 \\ 0.5x_1 - 0.2x_2 + 2x_3 - 0.5 \end{pmatrix},$$

$K = \{x \in \mathbb{R}^3 | x_1^2 + 0.4x_2^2 + 0.6x_3^2 \leq 1\}$,

$x_0 = [0, 1, 1]$,

Solution $= [1, 0, 0]$.

Table 3.1: Comparison of FRPM vs IAPRPM

Example	FRPM iterations	IAPRPM iterations
1	7	3
2	6	3
3	5	3
4	5	4
5	7	3
6	7	3
7	9	10

In most of the examples, the IAPRPM takes fewer number of iterations than FRPM. The reason could be that the half-spaces based on support points w^k are better approximations of K as compared to those based on the iterates x^k. Hence the iterates are closer to K. We find the support point only in the case when $x^k \notin K$ since otherwise the line segment connecting the anchor point and the current iterate would be inside K. It could be useful if we can improve the approximating halfspace for the case when $x^k \in K$ as well.

3.2 A gap function based algorithm

Both the IAPRPM and FRPM algorithms require Assumtion F3 to hold. Although these algorithms are easy to implement, the assumption may not hold for general variational inequality problems. To that end, the first part of this section analyses a gap function which can be evaluated using either of the two RPMs.

3.2.1 The gap function

Let us recall the generic gap function from Zhu and Marcotte's framework (Zhu and Marcotte, 1994). Let $\Omega(y, x) : K \times K \to \mathbb{R}$ be non-negative, continuously differentiable on $K \times K$, strongly convex on K with respect to y for all $x \in K$ and satisfy

$$\Omega(x, x) = 0 \text{ and } \nabla_y \Omega(x, x) = 0 \quad \forall\, x \in K. \tag{3.40}$$

Define

$$h(y, x) = \langle F(x), x - y \rangle - \Omega(y, x), \tag{3.41}$$

$$g(x) = \max_{y \in K} h(y, x) = h(H(x), x), \tag{3.42}$$

where $H(x)$ is the unique maximizer of $h(y, x)$ over K. Since the problem of finding $g(x)$ is to maximize a strongly convex objective function over a convex set, $H(x)$ is unique.

In this section, we assume K is a compact, convex set. Note that the variables in most applications have some natural upper and lower bounds, so it is not too restrictive to make such an assumption. Consider the following choice of $\Omega : K \times K \to \mathbb{R}$,

$$\Omega(u, v) = \frac{1}{2}(u - v)^T G(u - v), \tag{3.43}$$

where G is a positive definite matrix. Proposition 3.16 shows that this choice of Ω satisfies the conditions in (3.40). If we further choose $G = 2MI$, where I is an $n \times n$ identity matrix and

$$M = \max_{x \in K} \|F(x)\| + 1,$$

then the AVI (2.20) can be solved by any of the relaxed projection methods, see Proposition 3.17 and the comments following it.

Proposition 3.16. *For a positive definite matrix G,*

$$\Omega(y, x) = \langle y - x, G(y - x)\rangle$$

satisfies (3.40)*.*

Proof. Note that

$$\nabla_y \Omega(y, x) = 2G(y - x)$$

and

$$\nabla_x \Omega(y, x) = 2G(x - y).$$

Hence

$$\langle \nabla_y \Omega(y, x) + \nabla_x \Omega(y, x), y - x\rangle = 0.$$

□

With $\Omega(y, x) = M\langle y - x, y - x\rangle$, the $AVI(x)$ takes the following form

$$\langle F(x) + 2M(y - x), w - y\rangle \geq 0 \quad \forall w \in K. \tag{3.44}$$

Let $S(y) = F(x)+2M(y-x)$. For a fixed $x \in K$, $S(y)$ is the function that defines the AVI (3.44).

Proposition 3.17. *$S(y)$ satisfies* (3.13) *with $z = 0$ and D as the closed ball of radius $r = \max\{1, 2\|x\|_2\}$ around the origin.*

Proof. Consider $y \in \mathbb{R}^n$ such that $\|y\|_2 > \max\{1, 2\|x\|_2\}$. We have

$$\begin{aligned}
\langle S(y), y\rangle &= \langle F(x) + 2M(y - x), y\rangle \\
&= \langle F(x), y\rangle + M\|y\|_2^2 + M\|y\|_2^2 - 2M\langle x, y\rangle \\
&\geq -(M - 1)\|y\|_2 + M\|y\|_2^2 + M\|y\|_2^2 - 2M\langle x, y\rangle,
\end{aligned}$$

by the definition of M. Since $\|y\|_2 > 1$, $-\|y\|_2 \geq -\|y\|_2^2$. Also note that the fact $\|x\|_2 \leq \|y\|_2/2$ implies that last two terms in the above inequality add to a non-negative number. Hence, we have

$$\langle S(y), y\rangle \geq \|y\|_2^2 \geq \|y\|_2. \tag{3.45}$$

Since the only variable term in $S(y)$ is $2My$, $\alpha = 1$ satisfies (3.13) for the AVI. □

Proposition 3.17 shows that we can use the relaxed projection method to solve the $AVI(x^k)$. Note that the defining function $S(y) = F(x) + 2M(y - x)$, for $AVI(x)$, is zero at

$$O(x) = x - \frac{F(x)}{2M}.$$

Since $M > \max_{x\in K} \|F(x)\|$, $O(x)$ usually is close to x. If $c(O) \leq 0$, then $O(x) \in K$ and no iterative procedure is required to solve AVI(x), and $H(x) = O(x)$. In this case, the value of $g(x)$ can be found as,

$$\begin{aligned} g(x) &= \langle F(x), x - O(x)\rangle - M\langle O(x) - x, O(x) - x\rangle \\ &= \langle F(x), \frac{-F(x)}{2M}\rangle - M\langle \frac{F(x)}{2M}, \frac{F(x)}{2M}\rangle \\ &= \frac{\|F(x)\|_2^2}{4M}. \end{aligned} \tag{3.46}$$

If x^1 and x^2 are two points such that $x^2 = H(x^1) = O(x^1)$ and $O(x^2) \in K$, then we have

$$\begin{aligned} g(x^1) &= \frac{\|F(x^1)\|_2^2}{4M} \\ g(x^2) &= \frac{\|F(x^2)\|_2^2}{4M} \\ &= \frac{\|F(x^1 - \frac{F(x^1)}{2M})\|_2^2}{4M}. \end{aligned} \tag{3.47}$$

If in addition to being strongly monotone, $F(x)$ is also twice continuously differentiable, then from an application of Exercise 12.3-9 of (Ortega and Rheinboldt, 1970), we have

$$g(x^2) \leq \lambda g(x^1) \tag{3.48}$$

for some $\lambda < 1$ provided $O(x^1)$ and $O(x^2)$ are feasible. With the given choice of Ω, the following algorithm, which is a modification of the Zhu and Marcotte's general framework can be proved to converge to a solution of VI(K, F).

3.2.2 The algorithm

The following algorithm employs the gap function described in the preceeding subsection. Either of the RPMs can be used to solve the sub problems (AVIs) provided $0 \in K$. This is not a restrictive assumption since it can be achieved by a translation of the coordinate system. This algorithm is a modification of the algorithm described in (Zhu and Marcotte, 1994).

Algorithm 3.2 Descent algorithm

Initialization: Select $x^0 \in K$, $\gamma, \beta, \sigma \in (0,1)$. Set $k = 1$, loop $= 0$.

while loop = 0 **do**

if $x^k \in$ SOL(K, F) **then**

Set loop = 1.

else

if $O(x^k) = x^k - \frac{F(x^k)}{2M} \in K$ **then**

Set $x^{k+1} = O(x^k)$.

else

Solve AVI(x^k) for $H(x^k)$. Define $d^k = H(x^k) - x^k$.

if $g(x^k + d^k) < \gamma g(x^k)$ **then**

Set $x^{k+1} = x^k + d^k = H(x^k)$.

else

Select the smallest integer m such that

$$g(x^k) - g(x^k + \beta^m d^k) \geq -\sigma\beta^m \langle \nabla g(x^k), d^k \rangle,$$

and set $x^{k+1} = x^k + \beta^m d^k$.

end if

end if

Set $k = k + 1$.

end if

end while

Theorem 3.18. *If F is strongly monotone, the above algorithm generates a sequence of iterates that converges to a solution of VI(K, F).*

Proof. If there is a finite subsequence of consecutive iterates, say $\{x^r, x^{r+1}, \ldots, x^l\}$, with the $AVI(x^k)$ solved at the zero of $S(y)$ (that is, with $x^{k+1} = O(x^k)$). Then, for $k = r, r+1, \ldots, l-1$,

we have

$$g(x^{k+1}) \leq \lambda g(x^k)$$

for some $\lambda \in (0, 1)$. If there are an infinite number of such finite subsequences, we can combine them to get an infinite sub sequence of iterates for which the gap function value converges to zero.

Rest of the proof is the same as that of Theorem 6.1 in (Zhu and Marcotte, 1994). □

I have conducted tests on two small examples to check the effectiveness of this algorithm. I used the RPM of (Fukushima, 1986) to solve the AVIs. The numerical experiments showed a shortcoming in Fukushima's algorithm. It does not converge in practice if the solution to the VI(K, F) is not at the boundary of K. That is, when $F(x^*) = 0$. The problem is possibly due to the fact that division by $\|F\|$, which can be a very small number in the neighborhood of a zero of F, may cause inaccuracy in the computation. The IAPRPM overcomes this hurdle by first checking if an iterate is a zero of F.

If the zero of $S(y)$ is not feasible, we can start with the origin as the initial guess for solving $AVI(x^k)$ with either of the RPMs. We can also start with $x^0 = 0$ as the first iterate for Algorithm 3.2. If the initial guess is not close to the boundary of set K, the iterates moves quickly towards the boundary, without the need for solving the $AVI(x^k)$ with fukushima's method since the zero of $S(y)$, being close to the current iterate, is feasible for the first few iterations.

I conducted the numerical tests on the following two examples.

Example 3.8. *This example is used for testing the RPM in (Fukushima, 1986).*

$$\begin{aligned} F &= \begin{pmatrix} 2x_1 + 0.2x_1^3 - 0.5x_2 + 0.1x_3 - 4 \\ -0.5x_1 + x_2 + 0.1x_2^3 + 0.5 \\ 0.5x_1 - 0.2x_2 + 2x_3 - 0.5 \end{pmatrix} \\ K &= \{x \in \mathbb{R}^3 | x_1^2 + 0.4 * x_2^2 + 0.6 * x_3^2 \leq 1\}. \end{aligned}$$

Example 3.9. *I created this simple two dimenional example for testing.*

$$\begin{aligned} F &= \begin{pmatrix} 3x_1 + x_2 - 5 \\ 2x_1 + 5x_2 - 3 \end{pmatrix} \\ K &= \{x \in \mathbb{R}^2 | x_1^2 + x_2^2 \leq 1\}. \end{aligned}$$

The solutions to these problems are $(1, 0, 0)$ and $(0.9890, 0.1480)$ respectively. I used the same parameters for both the problems: $\beta = 0.7, \sigma = 0.5, \gamma = 0.9, M = 20$. The stopping criteria used is $\|x^k - x^{k+1}\| \leq 0.00001$. The following table summarizes the test results.

Table 3.2: Results for the descent algorithm

	Example 1	example2
Number of iterations	164	77
Total AVI problems solved	443	163
AVI problems with S(y) = 0 at solution	11	40
Total time elapsed	3.429 s	0.291 s
Final value of the gap function	1.78 $\times 10^{-12}$	2 $\times 10^{-11}$

As the table shows, in both the examples no iterative procedure was required to solve the AVIs for first few iterations. The total number of AVIs solved is significantly more than the number of iterations since the $g(x^k + d^k) < \gamma g(x^k)$ condition of Algorithm 3.1 does not hold for many iterations.

Remark: Algorithm 3.1 proposed here has two levels of iterations; each main iteration requires solving an AVI, which is an iterative procdeure itself.

Remark on the choice of $\{\rho_k\}$: Recall that an iteration of the RPMs requires the skewed projection of $(x^k - \rho_k G^{-1} \frac{F(x)}{\|F(x)\|_2})$ on a half space. A popular choice for the sequence ρ_k is the simple harmonic sequence of the form $\rho_k = \frac{L}{k}$, where L is a positive constant. This sequence goes to zero fast, and for some problems it leads to very slow convergence as the iterates approach the solution. Therefore I used the sequence $\rho_k = \frac{3}{k^{0.8}}$ for implementing RPM. Sequences of the form $\rho_k = \frac{L}{k^p}$,

where $p \in (0, 1]$ converge to zero slower, and the series $\sum_{k=0}^{\infty} \frac{L}{k^p}$ diverges. Hence these sequences can be better for use in RPMs.

The strong monotonicity of F is crucial for the convegence of Zhu and Marcotte's framework. Proximal point method only requires monotonicity of F. The function $F_{c,x}(y)$ defined in Proposition 2.4 is strongly monotone. VI(K, $F_{c,x}$) is solved approximately in the proximal point method described in the previous chapter. Using Algorithm 3.1 described here, VI(K, $F_{c,x}$) can be solved exactly.

3.3 A game theory example

In this section, we describe implementation of the descent framework based algorithm to an oligopoly problem. We consider a single product pricing scenario among several firms supplying the same product. Let there be N firms which supply a product in a non-cooperative fashion. The price that market offers depends on the total quantity Q of the product in the market. Let $p(Q)$ be the price as an inverse function of total quantity. Also, let q_i be the quantity supplied by the i^{th} firm. The total quantity Q is

$$Q = \sum_{i=1}^{N} q_i.$$

A Nash equilibrium solution for the market is a set of quantities $q_1^*, q_2^*, \ldots, q_N^*$ such that each q_i^* is the optimal solution to the following problem for all $i = 1, 2, \ldots, N$:

$$\max_{q_i \geq 0} \{q_i p(q_i + Q_i^*) - f_i(q_i)\}, \tag{3.49}$$

where $Q_i^* = \sum_{j=1, j \neq i}^{N} q_j^*$ and $f_i(q_i)$ is the i^{th} firm's cost function.

If each $f_i()$ is convex and continuously differentiable, the inverse demand function $p(Q)$ is strictly decreasing and continuously differentiable, and the industry revenue curve $Qp(Q)$ is concave, then $q_1^*, q_2^*, \ldots, q_N^*$ is a Nash equilibrium if and only if it solves the VI($\mathbb{R}_+^n, F$)

$$\langle F(q^*), q - q^* \rangle \geq 0 \text{ for all } q \in \mathbb{R}_+^n$$

where

$$F_i(q) = f_i'(q_i) - p(Q) - q_i p'(Q). \tag{3.50}$$

This problem will be referred to as VI_{NE} in the rest of this section.

Lemma 3.19. *If each $f_i()$ is strongly convex and continuously differentiable for $i = 1, 2, \ldots, N$, the inverse demand function $p(Q)$ is strictly decreasing and continuously differentiable, and the industry revenue curve $Qp(Q)$ is concave, the set defining VI_{NE} can be expressed as a box in $\mathbb{R}^n$.*

Proof. If there is only one supplier in the market, say the i^{th} supplier, then her revenue function is

$$R_i(q_i) = q_i p(q_i) - f_i(q_i). \tag{3.51}$$

For a firm to stay in the market, $R_i(q_i) > 0$ for some q_i values. Since $q_i p(q_i)$ is concave and f_i is strongly convex, $R_i(q_i)$ is strongly concave for $q_i > 0$. Hence there exists a range $[r_i, s_i] \in \mathbb{R}$ such that

$$R_i(q_i) \geq 0 \text{ for } q_i \in [r_i, s_i]$$

with $R_i(r_i) = 0$ and $R_i(s_i) = 0$. Let $s = \{s_i\}$ be the vector of all the s_i for $i = 1, 2, \ldots, N$. Then in the case of several competitors, if supplier i supplies quantity $q_i > s_i$, her revenue is

$$\begin{aligned} R_i(q_i) &= q_i p(q_i + Q_i) - f_i(q_i) \\ &\leq q_i p(q_i) - f_i(q_i) \\ &\leq 0 \end{aligned}$$

where $Q_i = \sum_{j=1, j\neq i}^{N} q_j$, the first inequality follows from $p()$ being a decreasing function, and the second follows from the fact that $q_i > s_i$.

Hence none of the suppliers would want to supply more than s_i. Thus the box defined by

$$K = \{x \in \mathbb{R}^n_+ |\ 0 \leq x_i \leq s_i\}$$

is the effective set over which VI_{NE} is defined. □

Also, note that if there is only supplier i in the market, her revenue maximization problem is:

$$\max_{q_i \geq 0} \; R_i(q_i) = q_i p(q_i) - f_i(q_i) \tag{3.52}$$

Since $q_i p(q_i)$ is concave and f_i is strongly convex, $R_i(q_i)$ is strongly concave and the problem (3.52) has a unique solution. Let that solution be u_i. Then $R_i(u_i)$ is the maximum revenue any supplier can generate. Hence we can use $M = \|R(u)\|_2 + 1$ for Algorithm 3.1.

With regard to the application of algorithm 3.1 to this equilibrium problem, note that the function $S(y) = F(x^k) + 2M(y - x^k)$ defining the $AVI(x^k)$ for a given iterate x^k is both strongly monotone and Lipschitz contiuous since

$$\langle S(e) - S(f), e - f \rangle = \langle 2M(e - f), e - f \rangle = 2M\|e - f\|_2^2$$

and

$$\langle S(e) - S(f), S(e) - S(f) \rangle = \langle 2M(e - f), 2M(e - f) \rangle = 4M^2\|e - f\|_2^2.$$

Recall (2.10) for the convergence of standard projection method. If we can find a positive definite matrix G such that

$$(4M^2)^2 \lambda_{max}(G) < 2(2M)\lambda_{min}^2(G) \tag{3.53}$$

holds, then the $AVI(x^k)$ can be solved with a fixed point iteration. We can find several such matrices. An example would be a diagonal matrix with one entry being $9M^3$, and rest being $7M^3$ each. The numbers have to be the order of M^3 because of (3.53).

This analysis leads to a descent method for solving box constrained VIs. Note that for a VI(K, F) with K being box constraints, and F being strongly monotone, a gap function of the form

$$g(x) = \max_{y \in K} \langle F(x), x - y \rangle - c\langle y - x, y - x \rangle \tag{3.54}$$

can be evaluated using a fixed point iteration. Evaluating the gap function $g(x)$ is equivalent to solving the AVI

$$\langle F(x) + 2c(y - x), w - y \rangle \geq 0 \quad \forall w \in K. \tag{3.55}$$

The function $S(y) = F(x) + 2c(y - x)$ defining the AVI is an affine function in y, hence it is monotone and Lipschitz continuous with $2c$ and $4c^2$ being the respective constants. Hence if we can find a positive definite matrix G with

$$(4c^2)^2\lambda_{max}(G) < 2(2c)\lambda^2_{min}(G)$$

then the map

$$\Pi_{K,G}(y - G^{-1}S(y))$$

is a contraction, and the AVI can be solved with a fixed point iteration that requires taking projections on a box in $\mathbb{R}^n$. That is, if K is a box then there is no need to to choose $c = M = \|R(u)\|_2 + 1$.

The following algorithm formalizes the above discussion.

Algorithm 3.3 Descent method for box constrained VIs

Initialization: Select $x^0 \in K$, $c > 0$, a positive definite matrix G with

$$(4c^2)^2 \lambda_{max}(G) < 2(2c)\lambda^2_{min}(G).$$

Set $k = 0$, loop $= 0$.

while loop = 0 **do**

if $x^{k+1} \in SOL(K, F)$ **then**

Set loop = 1.

else

Find $F(x^k)$.

Find $O(x^k) = x^k - \frac{F(x^k)}{2c}$.

if $O(x^k) \in K$ **then**

Set $x^{k+1} = x^k$.

else

Solve AVI(x^k) using the fixed point iteration (2.10) with x^0 as the starting point.

Define $d^k = H(x^k) - x^k$, where $H(x^k)$ is the solution of AVI(x^k).

if $g(x^k + d^k) < \gamma g(x^k)$ **then**

Set $x^{k+1} = x^k + d^k = H(x^k)$.

else

Select the smallest integer m such that

$$g(x^k) - g(x^k + \beta^m d^k) \geq -\sigma\beta^m \langle \nabla g(x^k), d^k \rangle,$$

and set $x^{k+1} = x^k + \beta^m d^k$.

end if

end if

Set k = k + 1;

end if

end while

(Harker, 1984) describe an example with five firms and the following inverse price function

$$Q = 5000p^{-1.1} \text{ or, } p = 5000^{\frac{1}{1.1}} Q^{\frac{-1}{1.1}}. \tag{3.56}$$

The cost functions of the firms are of the form:

$$f_i(q_i) = c_i q_i + \frac{\beta_i}{1+\beta_i} L_i^{\frac{-1}{\beta_i}} q_i^{\frac{\beta_i+1}{\beta_i}}. \tag{3.57}$$

Values of the parameters are given in the following table.

Table 3.3: Parameters for cost functions

firm	c_i	L_i	β_i
1	10	5	1.2
2	8	5	1.1
3	6	5	1.0
4	4	5	0.9
5	2	5	0.8

Here $R_i(1) > 0$ for $i = 1, 2, \ldots, 5$. To find the ends s_i of the box, we need to solve $R_i(q_i) = 0$ for each firm. s_i can be found via a binary search between 1 and some big supply level B_i for which $R_i(B_i) < 0$. We can make B_i bigger if the initial guess does not work.

For this problem instance, the vector s is

$$[178.2339,\ 173.8586,\ 163.7603,\ 148.3378,\ 128.8948].$$

i^{th} component of the function F defining the VI problem is

$$\begin{aligned} F_i(q) &= f_i'(q_i) - p(\sum_{i=1}^{N} q_i) - q_i p'(\sum_{i=1}^{N} q_i) \\ &= c_i + L_i^{\frac{1}{\beta_i}} q_i^{\frac{1}{\beta_i}} - 5000^{\frac{1}{1.1}} (\sum_{i=1}^{N} q_i)^{\frac{-1}{1.1}} + \frac{5000^{\frac{1}{1.1}}}{1.1} q_i (\sum_{i=1}^{N} q_i)^{\frac{-2.1}{1.1}}. \end{aligned} \tag{3.58}$$

From (Harker, 1984), the solution to the given problem is

$$q^* = [36.9120,\ 41.8420,\ 43.7050,\ 42.6650,\ 39.1820].$$

Since the VI is solved over the whole $\mathbb{R}^5_+$ in (Harker, 1984), the solution should either make $F = 0$ or be on the boundary. This solution indeed gives $F(q^*) = 0$ and the nash equilibrium profit values are

$$[548.2253,\ 443.0613,\ 345.6400,\ 262.7448,\ 196.3465].$$

Algorithm 3.3 converges to the solution in 33 iterations with $c = 0.5$. The algorithm in (Harker, 1984) converges in 20 iterations. But with algorithm 3.3, all the 33 iterations involve finding $O(x^k)$. The fixed point iteration is not required at all. Whereas the algorithm in (Harker, 1984) requires solving N, in this example 5, nonlinear optimization problems in every iteration.

Remark: The number of iterations required for the the convergence of Algorithm 3.3 depends on c. If c is taken to be a very large number then $O(x^k)$ might be too close to x^k, thereby increasing the number of iterations. If c is chosen to be too small, $O(x^k)$ might be outside K, thereby requiring the fixed point iteration for solving $AVI(x^k)$. In choosing c, it is better to err on the larger side, because then most iterations will only require finding $O(x^k)$, which is an easy computation to do.

CHAPTER 4

Local convergence result for relaxed projection methods

In last two chapters, we discussed two relaxed projection methods, and an interior anchor point relaxed projection method. The objective of this chapter is to complement the results for these RPMs by conducting a local convergence analysis. The major assumptions we use are that the solution satisfies the linear independence constraint qualification, and that the Jacobian matrix of $F(x)$ at the solution be positive definite. The results in this chapter do not require $F(x)$ to be globally monotone.

Here, we use the set defined by all of the linearized constraints as an outer approximation of K, as in (Fukushima, 1983; Taji and Fukushima, 1996). In each iteration, we compute the projection of some point onto that set. As a consequence, the algorithm considered here is slightly different from those discussed in the previous chapters. We show that the method converges linearly, when the starting point is sufficiently close to the solution and the step length t_k (defined below) is sufficiently small. We also give a formula for computing an upper bound for t_k. Our numerical experiments show that the upper bound given by the formula tends to be too conservative; in practice, one could use t_k bigger than that bound and still have fast convergence. The main purpose of that formula is to provide insights and guidance on step length selection.

In the next section, we introduce the Algorithm and discuss its termination condition. Section 3 analyzes the behavior of the (skewed) Euclidean projector onto the outer approximation set, and proves local convergence of the algorithm. Section 4 contains numerical examples and discussion.

4.1 The algorithm

To define the algorithm, we need to introduce some notation. Let the set K be defined by m constraints:

$$K = \{x \in \mathbb{R}^n \mid g_i(x) \leq 0, i = 1, \cdots, m\}. \tag{4.1}$$

Throughout this chapter we assume each g_i be a convex C^2 function from $\mathbb{R}^n$ to $\mathbb{R}$. The latter assumption ensures that K is convex. We also assume that K be nonempty. We use the notation $N_K(x)$ to denote the normal cone to K at x. Recall that the variational inequality problem is to find a point $x \in K$, such that

$$0 \in F(x) + N_K(x). \tag{4.2}$$

Given any point $x' \in \mathbb{R}^n$, we define a polyhedral convex set

$$K(x') := \{x \in \mathbb{R}^n \mid g_i(x') + \langle \nabla g_i(x'), x - x' \rangle \leq 0, i = 1, \cdots, m\}.$$

Note that the inclusion

$$K \subset K(x')$$

holds for every $x' \in \mathbb{R}^n$ under the convexity assumption. Moreover, $x' \in K(x')$ if and only if $x' \in K$. (The proofs are similar to Lemmas 3.1 and 3.2) Finally, if $x' \in K$ and a constraint qualification holds at x', then the tangent cone to K at x', denoted by $T_K(x')$, can be written as

$$T_K(x') := \{v \in \mathbb{R}^n \mid \langle \nabla g_i(x'), v \rangle \leq 0, i \in I(x')\},$$

where $I(x')$ denotes the set of active constraints at x'. Note that, for each such x', the set $x' + T_K(x')$ locally coincides with $K(x')$ around x'.

Algorithm 4.1 Local algorithm

Initialization: Select x^0, a sequence $\{t_k\}$ and a symmetric positive definite matrix G, and set $k = 1$, loop = 0.

while loop = 0 **do**

if $x^k \in \text{SOL}(K, F)$ **then**

Set loop = 1.

else

Set $x_{k+1} = \Pi_{K(x_k),G}(x_k - t_k G^{-1} F(x_k))$.

Set $k = k + 1$.

end if

end while

Clearly, the algorithm terminates at a point x^* if it satisfies

$$x^* = \Pi_{K(x^*),G}(x^* - t^* G^{-1} F(x^*)) \tag{4.3}$$

with t^* being the step length at that iteration. Note that it is not hard to see that (4.3) holds if and only if

$$-F(x^*) \in N_{K(x^*)}(x^*).$$

The following Lemma further relates (4.3) to the variational inequality (4.2), and shows that the algorithm terminates if and only if it finds a solution of (4.2).

Lemma 4.1. *If x^* is a solution to the variational inequality* (4.2) *and a constraint qualification holds at x^*, then* (4.3) *holds for any $t^* > 0$. Conversely, if a point $x^* \in \mathbb{R}^n$ satisfies* (4.3) *with some $t^* > 0$, then it is a solution to* (4.2)*.*

Proof. First, suppose x^* be a solution to (4.2) at which a constraint qualification holds. By the remarks at the beginning of this section, $x^* + T_K(x^*)$ locally coincides with $K(x^*)$. It follows that the tangent cone to $K(x^*)$ at x^* is exactly $T_K(x^*)$, and that the normal cone to $K(x^*)$ at x^* is $N_K(x^*)$. Since x^* solves (4.2), the vector $-F(x^*)$ belongs to $N_K(x^*)$, so it also belongs to the

normal cone to $K(x^*)$ at x^*. This proves (4.3) in view of the remarks right before the Lemma 4.1.

For the converse direction, suppose x^* satisfies (4.3) with some $t^* > 0$. This implies that $x^* \in K(x^*)$, and that $-F(x^*)$ belongs to $N_{K(x^*)}(x^*)$. The fact $x^* \in K(x^*)$ implies $x^* \in K$. The fact that $K \subset K(x^*)$ implies that $N_K(x^*)$ contains $N_{K(x^*)}(x^*)$, so $-F(x^*)$ belongs to $N_K(x^*)$. $\square$

4.2 Local convergence

To prepare for the convergence analysis of the algorithm, we first analyze the behavior of the skewed Euclidean projector onto the set $K(x')$ as x' varies. To this end, let G be a symmetric positive definite matrix, and consider the problem of finding $\Pi_{K(x'),G}(y)$, namely,

$$\min_{x \in K(x')} \frac{1}{2}(y-x)^T G(y-x). \tag{4.4}$$

Since $K(x')$ is a nonempty, closed and convex set for each $x' \in \mathbb{R}^n$, the problem (4.4) has a unique solution, denoted by $x(y, x')$ for each x' and y in $\mathbb{R}^n$. We will show that the function $x(y, x')$ is B-differentiable at certain points, and will give a formula for its B-derivatives. The following definition of B-differentiability follows from (Robinson, 1991).

Definition 4.1. A function $\Lambda : \mathbb{R}^n \to \mathbb{R}^m$ is B-differentiable at a point $z \in \mathbb{R}^n$, if there exists a positively homogeneous function $d\Lambda(z) : \mathbb{R}^n \to \mathbb{R}^m$, such that

$$\lim_{v \to 0} \frac{\Lambda(z+v) - \Lambda(z) - d\Lambda(z)(v)}{\|v\|} = 0.$$

Note that, because $K(x')$ is defined by linear constraints, the problem (4.4) is equivalent to its first-order necessary conditions:

$$\begin{aligned} &Gx - Gy + \sum_{i=1,\cdots,m} \nabla_x g_i(x')\lambda_i = 0, \\ &0 \le \lambda_i \perp g_i(x') + \langle \nabla g_i(x'), x - x' \rangle \le 0, i = 1, \cdots, m. \end{aligned} \tag{4.5}$$

If we define a function $f : \mathbb{R}^{3n+m} \to \mathbb{R}^{n+m}$ by

$$f(x, \lambda, y, x') = \begin{bmatrix} Gx - Gy + \sum_{i=1,\cdots,m} \nabla g_i(x')\lambda_i \\ -g(x') - \nabla g(x')(x - x') \end{bmatrix},$$

where $\nabla g(x')$ denotes the Jacobian matrix of g at x', then (4.5) can be equivalently written as a variational inequality with (x, λ) being the variable, and (y, x') being the parameter:

$$0 \in f(x, \lambda, y, x') + N_{\mathbb{R}^n \times \mathbb{R}^m_+}(x, \lambda). \tag{4.6}$$

The equivalence between (4.5) and (4.6) follows from the fact that the normal cone $N_{\mathbb{R}^n \times \mathbb{R}^m_+}(x, \lambda)$ is exactly the collection of vectors $(0, w) \in \mathbb{R}^n \times \mathbb{R}^m$ such that $0 \leq \lambda_i \perp w_i \leq 0$ for each $i = 1, \cdots, m$.

Now suppose that x^* be a solution to (4.2), which satisfies the linear independence constraint qualification (abbreviated as LICQ). In the following theorem, we apply the sensitivity analysis technique in (Robinson, 1995) to (4.6), to give a formula for the B-derivative of the function x at $(x^* - t^* G^{-1} F(x^*), x^*)$ for each $t^* \geq 0$. By assumption, x^* solves (4.2) and satisfies the LICQ, so it follows from Lemma 4.1 that

$$x^* = x(x^* - t^* G^{-1} F(x^*), x^*).$$

Moreover, there exists a unique vector $\lambda(t^*)$ in $\mathbb{R}^m$ such that $(x, \lambda, y, x') = (x^*, \lambda(t^*), x^* - t^* G^{-1} F(x^*), x^*)$ satisfies (4.5). The function $\lambda(t^*)$ is positively homogeneous; if we write $\lambda_1^* = \lambda(1)$ then $\lambda(t^*) = t^* \lambda_1^*$.

Theorem 4.2. *Let x^* be a solution to* (4.2) *which satisfies the LICQ, G be a symmetric positive definite $n \times n$ matrix, and $t^* \geq 0$. Define*

$$y^* = x^* - t^* G^{-1} F(x^*),$$

and write $\lambda^* = t^* \lambda_1^*$, with λ_1^* being as defined above. Partition the index set $\{1, \cdots, m\}$ into three

subsets,

$$I_1 = \{i : g_i(x^*) < 0\}\ , I_{00} = \{i : g_i(x^*) = 0, \lambda_i^* = 0\}\ \textit{and}\ I_{01} = \{i : g_i(x^*) = 0, \lambda_i^* > 0\},$$

and define

$$K_x = \left\{ v \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \langle \nabla g_i(x^*), v\rangle = 0 \textit{ if } i \in I_{01} \\ \langle \nabla g_i(x^*), v\rangle \leq 0 \textit{ if } i \in I_{00} \end{array} \right\}. \tag{4.7}$$

The function x is B-differentiable at (y^*, x^*), and its B-derivative for the direction (u, v') is given by

$$dx(y^*, x^*)(u, v') = \Pi_{K_x, G}(u - \sum_{i=1,\cdots,m} \lambda_i^* G^{-1} \nabla_{xx}^2 g_i(x^*) v'). \tag{4.8}$$

Proof. Let L be the Jacobian matrix of f with respect to (x, λ) at $(x^*, \lambda^*, y^*, x^*)$, namely,

$$L = \begin{bmatrix} G & \nabla g(x^*)^T \\ -\nabla g(x^*) & 0 \end{bmatrix},$$

and let $\bar{K}$ be the critical cone to $\mathbb{R}^n \times \mathbb{R}^m_+$ at (x^*, λ^*) associated with $-f(x^*, \lambda^*, y^*, x^*)$. Then

$$\begin{aligned} \bar{K} &= \{(v, w) \in T_{\mathbb{R}^n \times \mathbb{R}^m_+}(x^*, \lambda^*) \mid \langle f(x^*, \lambda^*, y^*, x^*), (v, w)\rangle = 0\} \\ &= \mathbb{R}^n \times \{w \in T_{\mathbb{R}^m_+}(\lambda^*) \mid \langle g(x^*), w\rangle = 0\} \\ &= \mathbb{R}^n \times \left\{ w \in \mathbb{R}^m \;\middle|\; \begin{array}{l} w_i = 0 \text{ if } g_i(x^*) < 0 \\ w_i \in \mathbb{R} \text{ if } g_i(x^*) = 0 \text{ and } \lambda_i^* > 0 \\ w_i \geq 0 \text{ if } g_i(x^*) = 0 \text{ and } \lambda_i^* = 0 \end{array} \right\} \\ &= \mathbb{R}^n \times \left\{ w \in \mathbb{R}^m \;\middle|\; \begin{array}{l} w_i = 0 \text{ if } i \in I_1 \\ w_i \in \mathbb{R} \text{ if } i \in I_{01} \\ w_i \geq 0 \text{ if } i \in I_{00} \end{array} \right\}. \end{aligned}$$

In order to apply the technique in (Robinson, 1995), let us define the normal map induced by the linear map L on $\bar{K}$ as

$$L_{\bar{K}}(v', w') := L(\Pi_{\bar{K}}(v', w')) + (v', w') - \Pi_{\bar{K}}(v', w').$$

The LICQ assumption implies that the normal map $L_{\bar{K}}$ is a homeomorphism from $\mathbb{R}^{n+m}$ to $\mathbb{R}^{n+m}$ (see (Robinson, 1980, Theorem 3.1)). Applying (Robinson, 1995, Theorem 3) to (4.6), we find neighborhoods X of x^*, Λ of λ^*, Y of y^* and X' of x^*, such that for each $y \in Y$ and $x' \in X'$, the variational inequality (4.6) has a unique solution in $X \times \Lambda$, which we denote by $h(y, x')$. Note that the x-component of $h(y, x')$ is exactly $x(y, x')$. Accordingly, we can write $h(y, x') = (x(y, x'), \lambda(y, x'))$. By (Robinson, 1995, Theorem 4), the function $h(y, x')$ is B-differentiable at (y^*, x^*), with the B-derivative $dh(y^*, x^*)$ being given by

$$\begin{aligned} dh(y^*, x^*)(u, v') &= \Pi_{\bar{K}}(L_{\bar{K}})^{-1}(-d_{y,x'} f(x^*, \lambda^*, y^*, x^*)(u, v')) \\ &= \Pi_{\bar{K}}(L_{\bar{K}})^{-1} \begin{bmatrix} Gu - \sum_{i=1,\cdots,m} \lambda_i^* \nabla^2_{xx} g_i(x^*) v' \\ 0 \end{bmatrix}. \end{aligned} \tag{4.9}$$

To prove (4.8), consider the following nonlinear program with u being the parameter and v being the variable:

$$\min \frac{1}{2}(u - v)^T G(u - v), \ \text{ s.t. } v \in K_x, \tag{4.10}$$

where K_x is defined in (4.7). Because K_x is a polyhedral convex set, v solves (4.10) if and only if there exist multipliers $\lambda_i, i \in I_{01} \cup I_{00}$, such that the following first-order conditions hold:

$$\begin{aligned} &Gv - Gu + \sum_{i \in I_{01} \cup I_{00}} \lambda_i \nabla g_i(x^*) = 0, \\ &\langle \nabla g_i(x^*), v \rangle = 0, i \in I_{01}, \\ &\lambda_i \geq 0 \perp \langle \nabla g_i(x^*), v \rangle \leq 0, i \in I_{00}. \end{aligned} \tag{4.11}$$

Define additionally $\lambda_i = 0$ for $i \in I_1$. Using the expression of $\bar{K}$ at the beginning of this proof, we may rewrite (4.11) as

$$\begin{bmatrix} -Gv + Gu - \sum_{i=1,\cdots,m} \lambda_i \nabla g_i(x^*) \\ \nabla g(x^*) v \end{bmatrix} \in N_{\bar{K}}(v, \lambda). \tag{4.12}$$

Now, define an affine map $\hat{L} : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$ by

$$\hat{L}(v, \lambda) = \begin{bmatrix} Gv - Gu + \sum_{i=1,\cdots,m} \lambda_i \nabla g_i(x^*) \\ -\nabla g(x^*)v \end{bmatrix},$$

to further rewrite (4.12) as

$$0 \in \hat{L}(v, \lambda) + N_{\bar{K}}(v, \lambda).$$

This is equivalent to

$$\begin{bmatrix} v \\ \lambda \end{bmatrix} = \Pi_{\bar{K}}(\hat{L}_{\bar{K}})^{-1}(0).$$

Note that $\hat{L}(v, \lambda) = L(v, \lambda) + (-Gu, 0)$. It then follows from the definition of normal maps that $\hat{L}_{\bar{K}}(v, \lambda) = L_{\bar{K}}(v, \lambda) + (-Gu, 0)$. Accordingly,

$$(\hat{L}_{\bar{K}})^{-1}(0) = (L_{\bar{K}})^{-1}(Gu, 0).$$

We have so far shown that the solution v to (4.10), together with the associated multiplier λ, can be equivalently expressed as

$$\begin{bmatrix} v \\ \lambda \end{bmatrix} = \Pi_{\bar{K}}(L_{\bar{K}})^{-1}(Gu, 0).$$

Recall the formula for $dh(y^*, x^*)$ in (4.9). Write $dh(y^*, x^*) = (dx(y^*, x^*), d\lambda(y^*, x^*))$. We see that $dx(y^*, x^*)(u, v')$ is the solution of (4.10) with u replaced by $u - \sum_{i=1,\cdots,m} \lambda_i^* G^{-1} \nabla^2_{xx} g_i(x^*)v'$. This proves (4.8). □

Next, define a function $\hat{x} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ by

$$\hat{x}(x', t) = \Pi_{K(x'),G}(x' - tG^{-1}F(x')). \tag{4.13}$$

We have

$$\hat{x}(x', t) = x(x' - tG^{-1}F(x'), x').$$

By the chain rule, the function $\hat{x}$ is B-differentiable at (x^*, t^*) for any $t^* \geq 0$, with the B-derivative

given by

$$\begin{aligned} &d\hat{x}(x^*,t^*)(v,t) \\ =&\Pi_{K_x,G}(v - t^*G^{-1}\nabla F(x^*)v - tG^{-1}F(x^*) - \sum_{i=1,\cdots,m} t^*(\lambda_1^*)_i G^{-1}\nabla^2_{xx} g_i(x^*)v). \end{aligned} \tag{4.14}$$

The equation above will play an important role in the local convergence proof. Before we present that proof, we give the following basic lemma. Recall the definition of the G-norm on $\mathbb{R}^n$ from chapter 1.

Lemma 4.3. *Let M be a positive definite $n \times n$ matrix, I_n be the $n \times n$ identity matrix, and G be a symmetric positive definite $n \times n$ matrix as before. Let $\|G^{-1}M\|_G$ denote the G-norm of $G^{-1}M$, that is,*

$$\|G^{-1}M\|_G = \max_{x\in\mathbb{R}^n, x\neq 0} \frac{\|G^{-1}Mx\|_G}{\|x\|_G}.$$

Define a scalar

$$\mu := \min_{x\in\mathbb{R}^n, x\neq 0} \frac{x^T M x}{\|x\|_G^2}.$$

Then $\mu > 0$, and for each scalar t one has

$$\|I_n - tG^{-1}M\|_G \leq (1 + t^2\|G^{-1}M\|_G^2 - 2t\mu)^{1/2},$$

which is less than 1 when $t < 2\mu/\|G^{-1}M\|_G^2$.

Proof. The fact $\mu > 0$ follows directly from the positive definiteness of M. For any $x \in \mathbb{R}^n$, we have

$$\begin{aligned} \|x - tG^{-1}Mx\|_G^2 &= \|x\|_G^2 - 2tx^T Mx + t^2\|G^{-1}Mx\|_G^2 \\ &\leq (1 + t^2\|G^{-1}M\|_G^2 - 2t\mu)\|x\|_G^2, \end{aligned}$$

and the Lemma follows. □

The following theorem is the main result of this chapter.

Theorem 4.4. *Let x^* be a solution to* (4.2) *which satisfies the LICQ, and let G be a symmetric positive definite $n \times n$ matrix. Let λ_1^* be as defined before Theorem 4.2. Suppose that $\nabla F(x^*)$ be*

positive definite. Define

$$M := \nabla F(x^*) + \sum_{i=1,\cdots,m} (\lambda_1^*)_i \nabla_{xx}^2 g_i(x^*),$$

$$\mu := \min_{x \in \mathbb{R}^n, x \neq 0} \frac{x^T M x}{\|x\|_G^2},$$

and let t^* be a positive scalar satisfying $t^* < 2\mu / \|G^{-1}M\|_G^2$. Then there exists a neighborhood X_0 of x^* such that, the sequence $\{x_k\}$ will remain in X_0 and converge Q-linearly to x^*, whenever $x_0 \in X_0$ and $t_n = t^*$ for all n.

Proof. Let t^* be as given in the statement of the present theorem, and write $y^* = x^* - t^* G^{-1} F(x^*)$. Let ϵ be a positive scalar satisfying

$$\epsilon < 1 - (1 + (t^*)^2 \|G^{-1}M\|_G^2 - 2t^*\mu)^{1/2}.$$

Because the function $\hat{x}$ is B-differentiable at (x^*, t^*) with its B-derivative given in (4.14), there exist neighborhoods X_0 of x^* and T_0 of t^* such that

$$\|\hat{x}(x,t) - \hat{x}(x^*,t^*)\|_G \leq \|d\hat{x}(x^*,t^*)(x - x^*, t - t^*)\|_G + \epsilon \|x - x^*\|_G + \epsilon \|t - t^*\|,$$

for each $x \in X_0$ and $t \in T_0$. Without loss of generality, we may assume that X_0 be a ball around x^* under the G-norm.

Now, assume that $x_k \in X_0$. We have

$$\begin{aligned}
\|x_{k+1} - x^*\|_G &= \|\hat{x}(x_k, t^*) - \hat{x}(x^*, t^*)\|_G \\
&\leq \|d\hat{x}(x^*, t^*)(x_k - x^*, 0)\|_G + \epsilon \|x_k - x^*\|_G \\
&= \|\Pi_{K_x, G}(x_k - x^* - t^* G^{-1} M (x_k - x^*))\|_G + \epsilon \|x_k - x^*\|_G \\
&\leq \|x_k - x^* - t^* G^{-1} M (x_k - x^*)\|_G + \epsilon \|x_k - x^*\|_G \\
&\leq ((1 + (t^*)^2 \|G^{-1}M\|_G^2 - 2t^*\mu)^{1/2} + \epsilon) \|x_k - x^*\|_G.
\end{aligned}$$

This proves that x_{k+1} remains in X_0 and that the sequence $\{x_k\}$ converges Q-linearly to x^* with rate $(1 + (t^*)^2 \|M\|^2 - 2t^*\mu)^{1/2} + \epsilon$. □

Theorem 4.4 suggests, that for fast convergence, one shall choose t^* to minimize

$$(1 + (t^*)^2\|G^{-1}M\|_G^2 - 2t^*\mu)^{1/2}.$$

Clearly, the minimizer of the quantity above is

$$t^* = \mu/\|G^{-1}M\|_G^2, \tag{4.15}$$

and the minimum value achieved is

$$(1 - \mu^2/\|G^{-1}M\|_G^2)^{1/2}.$$

Thus, the algorithm converges fast locally near a solution, if $\mu/\|G^{-1}M\|_G$ is close to 1, for example, when $G = I_n$ and M is well-conditioned.

Note that one does not know the values of μ and $\|G^{-1}M\|_G$ before finding the solution x^*. Hence, in reality, one would not use the formula (4.15) to determine precisely an upper bound for the parameter t^*. The purpose of that formula is to serve as a general guideline for the choice of t^*.

4.3 Numerical examples

We use the following examples to illustrate the algorithm. For all the examples, we use identity matrices of appropriate sizes as the matrix G. Starting from a point close enough to the solution, we use the t^* values computed by (4.15). For comparison purposes, we also test some other choices of step lengths. We stop the algorithm when $\|x_{k+1} - x_k\| \leq 0.001$.

Example 4.1. *Let*

$$F(x) = \begin{pmatrix} 3x_1 + 4x_2 + 5 \\ 2x_1 + 5x_2 - 4 \end{pmatrix},$$

and let the set S be

$$K = \left\{ x \in \mathbb{R}^2 \;\middle|\; \begin{array}{r} x_1^2 + 4x_2^2 \leq 4 \\ 2x_1^2 + x_2^2 \leq 6 \\ 2x_1 + x_2 \geq -1 \end{array} \right\}.$$

The solution to the problem is $x^* = (-0.9412, 0.8824)$. *The first and the third constraints are binding at the solution. The multipliers for the solution are* $(\lambda_1^*)_1 = 0.5404$, $(\lambda_1^*)_2 = 0$ *and* $(\lambda_1^*)_3 = 2.344$. *The formula* (4.15) *gives* $t^* = 0.0296$. *Starting from the initial point* $(-0.8, 0.8)$ *and letting* $t_k = t^*$ *for all* k, *the algorithm converges to the solution in 3 iterations. The choice* $t_k = 1/k$ *also gives convergence in three iterations from the same starting point. Indeed, one can choose* t_k *to be any constant value in the set* $\{0.001, 0.002, \ldots, 0.999, 1\}$, *and still find convergence in three iterations.*

Example 4.2. *This example is adapted from the example in (Fukushima, 1986). Let*

$$F(x) = \begin{pmatrix} 2x_1 + 0.2x_1^3 - 0.5x_2 + 0.1x_3 - 4 \\ -0.5x_1 + x_2 + 0.1x_2^3 + 0.5 \\ 0.5x_1 - 0.2x_2 + 2x_3 - 0.5 \end{pmatrix},$$

and let the set S *be*

$$K = \left\{ x \in \mathbb{R}^3 \;\middle|\; \begin{array}{r} x_1^2 + 0.4x_2^2 + 0.6x_3^2 \leq 1 \\ 0.6x_1^2 + 0.4x_2^2 + x_3^2 \leq 1 \\ x_1 + x_2 + x_3 \geq \sqrt{3} \end{array} \right\}.$$

The solution to the problem is $x^* = (0.9168, 0.4850, 0.3303)$. *The first and the third constraints are binding at the solution. The multipliers for the solution are* $(\lambda_1^*)_1 = 1.9091$, $(\lambda_1^*)_2 = 0$ *and* $(\lambda_1^*)_3 = 1.2787$. *The formula* (4.15) *gives* $t^* = 0.0629$. *Starting from the initial point* $(0.9, 0.48, 0.33)$, *the algorithm converges to the solution in 2 iterations. The choice* $t_k = 1/k$ *gives convergence in 4 iterations.*

Example 4.3. *This example is modified from the Suzuki Rosen problem in (Rosen and Suzuki, 1965). We changed the function* F *in that problem so that* ∇F *is positive definite. Let*

$$F(x) = Ax - b,$$

where

$$A = \begin{pmatrix} 3.0006 & 0.0212 & 0.0141 & 0.0215 & 0.0088 \\ 0.0212 & 3.7093 & 0.4708 & 0.7193 & 0.2930 \\ 0.0141 & 0.4708 & 4.3125 & 0.4775 & 0.1945 \\ 0.0215 & 0.7193 & 0.4775 & 3.7295 & 0.2971 \\ 0.0088 & 0.2930 & 0.1945 & 0.2971 & 3.1210 \end{pmatrix},$$

and $b = (-1.5849, 15.8236, 13.1763, 12.0172, 138.7089)$. *The set* S *is defined as*

$$K = \left\{ x \in \mathbb{R}^5 \;\middle|\; \begin{array}{r} x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4 + x_5 \leq 0 \\ 4x_1^2 + 4x_2^2 + 5x_3^2 + 4x_4^2 - 2x_1 - 8x_2 - 18x_3 + 4x_4 + x_5 - 24 \leq 0 \\ 4x_1^2 + 7x_2^2 + 5x_3^2 + 7x_4^2 - 8x_1 - 5x_2 - 21x_3 + 4x_4 + x_5 - 30 \leq 0 \\ 7x_1^2 + 4x_2^2 + 5x_3^2 + x_4^2 + x_1 - 8x_2 - 21x_3 + 4x_4 + x_5 - 15 \leq 0 \\ -100 \leq x_i \leq 0 \end{array} \right\}.$$

The solution to the problem is $x^* = (0, 1, 2, -1, 44)$. *The first, second and the third constraints are binding at the solution. The multipliers for the solution are* $(\lambda_1^*)_1 = \frac{1}{3}$, $(\lambda_1^*)_2 = \frac{1}{3}$ *and* $(\lambda_1^*)_3 = \frac{1}{3}$, *and the remaining components of* (λ_1^*) *are zero. The formula* (4.15) *gives* $t^* = 0.0317$. *Starting from the initial solution* $(0, 1.2, 2, -1, 44.2)$, *the algorithm converges to the solution in 4 iterations. The sequence* $t_k = 1/k$ *gives convergence in 8 iterations.*

4.4 Conclusions

This chapter proves the local linear convergence of an outer approximation projection method for solving variational inequalities. Such convergence is guaranteed, if the function F has a positive definite Jacobian matrix at the solution, the LICQ holds there, and the step lengths are sufficiently small. Most existing projection based algorithms for solving variational inequalities use step lengths converging to zero. This leads to very small steps when the iterates are close to the solution. As our results show, the step lengths need not converge to zero. Using longer step lengths makes it possible to converge in fewer iterations.

In practice, one does not know in advance if the constraint qualification holds at the solution,

or if the iterate is close enough to the solution. As a general guideline, when two iterates are close and the Jacobian of F at the current iterate is positive definite, we can estimate t^* using the formula (4.15) with information from the current iterate, and use t^* as an upper bound for step lengths in the remaining iterations.

Appendix

The appendix contains the codes used for solving the examples using various algorithms. The programs have been written in a way that makes it possible to use only two functions to solve all the examples. Examples 1 to 7 correspond to those in section 3.1, and example 8 corresponds to the Oligopoly problem. Description of The functions are given in the following table.

Table A.1: Functions for implementing algorithms

FRPM1	Fukushima's RPM
IAPRPM1	IAPRPM
Gap_Algo	Gap function based Algorithm 3.2
Solve_VI_box	Gap function based algorithm for box constrained VIs
Solve_AVI_box	AVI for solving box constrained VIs
find_next_iterate	finds the next iterate using line search for Algorithm 3.2

The functions FRPM, IAPRPM1, Gap_Algo, Solve_VI_box call the other functions as needed. The codes are in next few pages.

```
% Fukushima's relaxed projection algorithm
function solution = FRPM1(example)
% Get the information about he exmaples
[num_var num_cons] = get_info(example);
M = get_M(example);
G = get_G(example);
[sigma beta gamma epsilon] = get_params(example);
% Get the starting point
curr_it = get_x0(example);

% Get the Function value
f_current = get_function_value(example, curr_it);

loop = 0;
iterations = 0;
rho = 3;
tic;
while (loop == 0)
        % Find the function value, constraint value, sub gradient
        f_curr = get_function_value(example, curr_it);
        tau = rho/norm(f_curr);
        c_curr = get_constr_LHS(example, curr_it);
        g_curr = get_subgradient(example, curr_it);
        % Findthe skewed projection
        lambda = max(0, (c_curr - tau*dot(f_curr, G*g_curr))/
                 dot(g_curr, G*g_curr));
        next_it = curr_it - G*(tau*f_curr + lambda*g_curr);
        % Check termination
        if (norm(next_it - curr_it) <= epsilon)
            loop = 1;
        end
        curr_it = next_it;
        % Update data
        iterations = iterations + 1;
        rho = rho/(iterations + 1)^(0.8);
```

```
end
toc;
iterations
solution = curr_it;
```

```
function solution = IAPRPM1(example)
% Get the information about he exmaples
[num_var num_cons] = get_info(example);
M = get_M(example);
G = get_G(example);
[sigma beta gamma epsilon] = get_params(example);
% Get the anchor point and the closeness parameter
% for the approximating hyperplane
[anchor closeness] = get_anchor(example);
% Get the starting iterate
curr_it = get_x0(example);


z = curr_it;
c_z = closeness + 1;
f_current = get_function_value(example, curr_it);


loop = 1;
iterations = 0;
rho = 3;
tic;
while (loop == 1)
        f_curr = get_function_value(example, curr_it);
        tau = rho/norm(f_curr);
        % Find the approximating hyperplane
        c_curr = get_constr_LHS(example, curr_it);
        if c_curr <= closeness
            z = curr_it;
            c_z = c_curr;
        else
            c_z = c_curr;
            end1 = anchor;
            end2 = curr_it;
            while c_z > closeness
               z = (end1 + end2)/2;
               c_z = get_constr_LHS(example, z);
               if c_z < 0
```

```
                end1 = z;
                c_z = c_curr;
            else
                if c_z > closeness
                    end2 = z;
                end
            end
        end
    end
    % Find the projection
    g_z = get_subgradient(example, z);
    to_be_proj = curr_it - tau*f_curr;
    lambda = max(0, (c_z - dot(z - to_be_proj, g_z))/
         dot(g_z, G*g_z));
    next_it = curr_it - G*(tau*f_curr + lambda*g_z);
    % Check for termination
    if (norm(next_it - curr_it) <= epsilon)
        loop = 0;
    end
    curr_it = next_it;
    iterations = iterations + 1;
    rho = rho/(iterations + 1);

end
toc;
iterations
solution = curr_it;
```

```
% Implementationof algorithm 3.2
function curr_it = Gap_Algo(example)

fid = fopen('exp.txt', 'w');

% Get the number of constraints, variables, and other parameters for the
% example
[num_var num_cons] = get_info(example);
M = get_M(example);
[sigma beta gamma epsilon] = get_params(example);
% Start with zero as the initial iterate
curr_it = zeros(num_var, 1) ;

outer_loop = 1;
num_outer_iterations = 0;
% marker for outer loop
inner_for_outer = 0;
while (outer_loop == 1)
    fprintf(fid,'Iteration number = %d \r\n',num_outer_iterations);
    fprintf(fid,'current iterate = [ ');
    fprintf(fid,'%6.6f  ',curr_it);
    fprintf(fid,'] \r\n ');
    % Get function value for the iterate
    f_curr = get_function_value(example, curr_it);
    fprintf(fid,'F current = [ ');
    fprintf(fid,'%6.6f  ', f_curr);
    fprintf(fid,'] \r\n ');
    % Solve the current AVI
    [current_AVI_solution num_inner_iterations] =
      solve_AVI(example, curr_it, f_curr, epsilon, M, fid);
    fprintf(fid,'current AVI solution = [ ');
    fprintf(fid,'%6.6f  ', current_AVI_solution);
    fprintf(fid,'] \r\n ');
    % Check termination condition.
    if (norm(curr_it - current_AVI_solution) <= epsilon)
            outer_loop = 0;
```

```
            curr_it = current_AVI_solution;
            fprintf(fid, 'outer loop zeroed since iterates
                are close at first check \r\n');
    % Set the next iterate
    else
            next_iterate = find_next_iterate(example, curr_it,
                           f_curr, current_AVI_solution, sigma,
                           beta, gamma, epsilon, M, fid);
            if (norm(curr_it - next_iterate)<= epsilon)
                outer_loop = 0;
                fprintf(fid, 'outer loop zeroed since iterates
                        are close at second check \r\n');
            end
            curr_it = next_iterate;
    end
    % Increment number of outer iteartions
    num_outer_iterations = num_outer_iterations + 1;
    % random test criteria
    if (num_outer_iterations == 1228)
        outer_loop = 0;
    end
    % Update the number of inner iterations
    inner_for_outer(num_outer_iterations) = num_inner_iterations;
    fprintf(fid, '_____ \r\n \r\n \r\n');
end
curr_it
fclose(fid);
```

```
% Function for solving box constrained VI
function nash = solve_VI_box(example)
% Get the example info
[num_var num_const] = get_info(example);
[sigma beta gamma epsilon] = get_params(example);
M = get_M(example);
q_current = zeros(1, num_var);
D = diag([7*M^3 7*M^3 7*M^3 7*M^3 9*M^3]);
D_inv = inv(D);
loop = 1;
tau = 3;
iterations = 0;
required_inner_loop = 0;
% The main loop
while (loop == 1)
    iterations = iterations + 1;
    % Find the solution of the current AVI, and if it is the
     zero for the AVI
    [at_zero solution] = solve_AVI_box(example, q_current)
    % Check termination
    if (norm(q_current - solution) <= epsilon)
        loop = 0;
        break;
    end
    % Check if the zero is feasible
    if (at_zero == 1)
        q_current = solution;
    else
        % Ensure that the descent condition holds

        % Find the function and gap function value
        F_current = get_function_value(example, q_current);
        g_current = dot(F_current,q_current - solution) +
        M*dot(q_current - solution, q_current - solution);

        % Find the gap function value for the current
```

```
        % solution to AVI
        [at_zero1 solution1] = solve_AVI_box(example, solution);
        F_candidate = get_function_value(example, solution);
        g_candidate = dot(F_candidate,solution - solution1) +
        M*dot(solution - solution1, solution - solution1);
        % First descent check
        if (g_candidate < gamma*g_current)
            q_current = solution;
        else
            % Line search
            inner_loop = 1;
            m = 1;
            del_F =  get_function_jac(example, q_current);
            direction = solution - q_current;
            del_g = F_current - direction*del_F - 2*M*direction;
            while (inner_loop == 1)
                candidate = q_current + beta * direction;
                [at_zero1 solution1] = solve_AVI_box(example,
                              candidate);
                g_candidate = dot(F_candidate,candidate -
                       solution1) + M*dot(candidate -
                      solution1, candidate - solution1);
                if (g_current - g_candidate >=
                -1*sigma*(beta^m)*dot(del_g, direction))
                   q_current = candidate;
                   inner_loop = 0;
                else
                   m = m + 1;
                end
            end
        end

    end
end
```

```
iterations
required_inner_loop
nash = q_current
```

```
% Function for solving the AVI for box constrained problem
function [at_zero solution] = solve_AVI_box(example, x)
epsilon = 0.00001;
[box_left box_right] = get_box(example);
[num_var num_const] = get_info(example);
M = get_M(example);
q_current = zeros(1, num_var) + 1;
D = 7*M^3 * eye(num_var);
D(1,1) = 9*M^3;
D_inv = inv(D);

% Find the Q and F values
F_current = get_function_value(example, q_current);
projection = 0;
% Find O(x^k)
zero_current = q_current - F_current/(2*M);
% If O(x^k) is feasible, it is the next iterate
if ((min(box_left <= zero_current) == 0) &&
    (min(box_right >= zero_current) == 0))
    solution = zero_current;
    at_zero = 1;
else
    at_zero = 0;
    % Else solve AVI with fixed point iteration
    inner_iterations = 0;
    inner_loop = 1;
    inner_current = x;
    % The fixed point iteration
    while (inner_loop == 1)
        inner_iterations = inner_iterations + 1;
        f_current = F_current + 2*M*(inner_current - q_current);
        to_be_projected = inner_current - (D_inv*f_current')';
        for j = 1:num_var
            if (to_be_projected(j) < box_left(j))
                projection(j) = box_left(j);
            elseif (to_be_projected(j) > box_right(j))
```

```
                projection(j) = box_right(j);
            else
                projection(j) = to_be_projected(j);
            end
        end
        if (norm(inner_current - projection) < epsilon)
        inner_loop = 0;
        end
        inner_current = projection;
    end
    solution = inner_current;
end
```

```
% Function for finding the next iterate via line search
function [next_iterate g_curr] = find_next_iterate(example,
curr_it, f_curr, current_AVI_solution, sigma, beta, gamma,
epsilon, M, fid)

% find direction
dir = current_AVI_solution - curr_it;
% Find f value for current candidate

% Find AVI soln for current candidate
[AVI_soln_for_current_candidate iters]= solve_AVI(example,
          current_AVI_solution, f_curr, epsilon, M, fid);
% find the current g value
g_curr = dot(f_curr, -dir) - M * dot(dir, dir);
% fprintf(fid,'g current = %6.6e \r\n',g_curr);
% find g value for the candidate
% find direction for candidate
dir_for_g_candidate = AVI_soln_for_current_candidate -
                current_AVI_solution;
g_candidate = dot(f_curr, -dir_for_g_candidate) -
    M * dot(dir_for_g_candidate, dir_for_g_candidate);
% fprintf(fid,'g candidate = %6.6e \r\n',g_candidate);
iterateion_here = 0;
% Check first descent condition
if (g_candidate < gamma*g_curr)
   next_iterate = current_AVI_solution;
else
   % Line search
   del_f =  get_function_jac(example, curr_it);

   del_g = f_curr - (del_f'* (dir)) - 2*M*dir;
   m = 1;
   loop = 1;
   while (loop == 1)
       iterateion_here = iterateion_here + 1;
       curr_candidate = curr_it + (beta^m) * dir;
```

```
        f_current_candidate = get_function_value(example,
                        curr_candidate);
        [AVI_soln_for_current_candidate iters] = solve_AVI
                  (example, curr_candidate,
                  f_current_candidate, epsilon, M, fid);
        dir_for_g_candidate = AVI_soln_for_current_candidate
                        - curr_candidate;
        g_candidate = dot(f_current_candidate, -
                  dir_for_g_candidate) -
                  M * dot(dir_for_g_candidate,
                  dir_for_g_candidate);
        if (g_curr - g_candidate >=
         -1*sigma*(beta^m)*dot(del_g, dir))
            next_iterate = curr_candidate;
            loop = 0;
        else
            m = m + 1;
        end
    end
end
```

Table A.2: Functions giving information about problem instances

get_info	number of variables and constarints for a given problem
get_M	M value for a given problem
get_params	γ, σ, and ϵ for a given problem
get_function_value	function values for a given problem
get_function_jac	function Jacobian for a given problem
get_constraint_LHS	$c(x)$ values for a given problem
get_subgradient	subgradient values for a given problem

```
% Function for supplying parameters. Example 8 is the oligopoly problem.
function [sigma beta gamma epsilon] = get_params(example)
epsilon = 0.00000001;
if example == 1
    sigma = 0.00002;
    beta = 0.3;
    gamma = 0.9998;
end

if example == 2
    sigma = 0.00002;
    beta = 0.3;
    gamma = 0.9998;
end

if example == 3
    sigma = 0.00002;
    beta = 0.3;
    gamma = 0.9998;
end

if example == 4
    sigma = 0.00002;
    beta = 0.3;
    gamma = 0.9998;
end

if example == 5
    sigma = 0.00002;
    beta = 0.3;
    gamma = 0.9998;
end

if example == 6
    sigma = 0.00002;
    beta = 0.3;
```

```
    gamma = 0.9998;
end

if example == 7
    sigma = 0.00002;
    beta = 0.3;
    gamma = 0.9998;
end

if example == 389
    sigma = 0.00002;
    beta = 0.3;
    gamma = 0.9998;
end

if example == 389.1
    sigma = 0.00002;
    beta = 0.3;
    gamma = 0.9998;
end

if example == 10
    sigma = 0.00002;
    beta = 0.3;
    gamma = 0.9998;
end

if example == 11
    sigma = 0.00002;
    beta = 0.3;
    gamma = 0.9998;
end

if example == 12
    sigma = 0.00002;
    beta = 0.3;
```

```
    gamma = 0.9998;
end
```

```
% Function for supplying M value
function M = get_M(example)
if example == 1
    M = sqrt(12);
end

if example == 2
    M = sqrt(45);
end

if example == 3
    M = sqrt(121*10);
end

if example == 4
    M = sqrt(21*21*20);
end

if example == 5
    M = sqrt(exp(3));
end

if example == 6
    M = sqrt(12);
end

if example == 7
    M = sqrt(21*21*20);
end

if example == 389
    A = [100    100 10  5   10  0   0   25  0   10  55  5   45  20  0;
         90 100 10  35  20  5   0   35  55  25  20  0   40  25  10;
         70 50  0   55  25  100 40  50  0   30  60  10  30  0   40];
    b = [385; 470; 560];
    coeffs = [486; 640; 758; 776; 0; 707; 175; 619; 627; 614;
```

```
           475; 377; 524; 468; 529]
    constts = [823; 823; 081; 239; 929; 022; 170; 097; 298;
            982; 871; 962; 653; 536; 523];
    M = 0;
    for i = 1:15
       mini = min([b(1)/A(1,i)   b(2)/A(2,i)  b(3)/A(3,i)])
       M = M + (mini*coeffs(i)+constts(i))^2;
    end
    M = sqrt(M);

    M = 100;
end

if example == 389.1
    A = [100    100 10  5   10  0   0   25  0   10  55  5   45  20  0;
         90 100 10  35  20  5   0   35  55  25  20  0   40  25  10;
         70 50  0   55  25  100 40  50  0   30  60  10  30  0   40];
    b = [385; 470; 560];
    coeffs = [486; 640; 758; 776; 0; 707; 175; 619; 627; 614;
           475; 377; 524; 468; 529]
    constts = [823; 823; 081; 239; 929; 022; 170; 097; 298;
            982; 871; 962; 653; 536; 523];
    M = 0;
    for i = 1:15
       mini = min([b(1)/A(1,i)   b(2)/A(2,i)  b(3)/A(3,i)])
       M = M + (mini*coeffs(i)+constts(i))^2;
    end
    M = sqrt(M);

    M = 100;
end

if example == 10
    M = 200;
end
```

```
if example == 11
    M = 0.5;
end


if example == 12
    M = 15;
end
```

```
% Function for supplying number of variables and constraints.
% Example 8 is the oligopoly problem.
function [num_var num_cons] = get_info(example)
if example == 1
    num_var = 3;
    num_cons = 1;
end
if example == 2
    num_var = 5;
    num_cons = 1;
end
if example == 3
    num_var = 10;
    num_cons = 1;
end
if example == 4
    num_var = 20;
    num_cons = 1;
end
if example == 5
    num_var = 3;
    num_cons = 1;
end
if example == 6
    num_var = 3;
    num_cons = 1;
end
if example == 7
    num_var = 20;
    num_cons = 1;
end
if example == 389
    num_var = 15;
    num_cons = 1;
end
if example == 389.1
```

```
    num_var = 15;
    num_cons = 1;
end
if example == 10
    num_var = 3;
    num_cons = 1;
end
if example == 11
    num_var = 5;
    num_cons = 1;
end
if example == 12
    num_var = 2;
    num_cons = 1;
end
```

```
% Function for supplying constraint LHS.
function return_value = get_constraint_LHS(example, point)
return_value = 0;
if example == 1
    for i = 1:3
        return_value = return_value + point(i)^2;
    end
    return_value = return_value - 1;
end

if example == 2
    for i = 1:5
        return_value = return_value + point(i)^2;
    end
    return_value = return_value - 2;
end

if example == 3
    for i = 1:10
        return_value = return_value + point(i)^2;
    end
    return_value = return_value - 5;
end
if example == 4
    for i = 1:20
        return_value = return_value + point(i)^2;
    end
    return_value = return_value - 15;
end

if example == 5
    for i = 1:3
        return_value = return_value + point(i)^2;
    end
    return_value = return_value - 1;
end
```

```
if example == 6
    for i = 1:3
        return_value = return_value + point(i)^2;
    end
    return_value = return_value - 1;
end

if example == 7
    for i = 1:20
        return_value = return_value + point(i)^2;
    end
    return_value = return_value - 15;
end
```

```
% Function for supplying jacobian of F. Example 8 is the oligopoly problem.
function return_value = get_function_jac(example, point)
return_value = 0;
if example == 1
    return_value = eye(3);
end

if example == 2
    return_value = eye(5);
end

if example == 3
    return_value = eye(10);
end

if example == 4
    return_value = eye(20);
end

if example == 5
    return_value = [e^point(1) 0 0; 0 e^point(2) 0;
                0 0 e^point(3)];
end

if example == 6
    return_value = [3*point(1)^2 0 0; 0 3*point(2)^2 0;
                0 0 3*point(3)^2];
end

if example == 7
    dgnl = zeros(20,1);
    for i = 1:20
        dgnl(i) = 3*point(i)^2;
    end
    return_value = diag(dgnl);
end
```

```
if example == 389
    return_value = diag([486; 640; 758; 776; 0; 707; 175;
              619; 627; 614; 475; 377; 524; 468; 529]);
end

if example == 389.1
    dgnl_c = [486; 640; 758; 776; 0; 707; 175; 619; 627;
           614; 475; 377; 524; 468; 529];
    dgnl = zeros(20,1);
    for i = 1:20
        dgnl(i) = 3*dgnl_c(i)*point(i)^2;
    end
    return_value = diag(dgnl);

end

if example == 11
    return_value = eye(5);
    const1 = (5000^(1/1.1));
    const2 = 2*const1/1.1;
    const3 = (2.1/1.1^2)*const1;
    b = [1.2 1.1 1.0 0.9 0.8];
    L = [5 5 5 5 5];
    c = [10 8 6 4 2];
    Q = sum(point);
    for i = 1:5
        for j = 1:5
            if (i == j)
                return_value(i,j) = L(i)^(1/b(i)) + const2 *
                            Q^(-2.1/1.1) - const3 *
                            point(i) * Q^(-3.2/1.1);
            else
                return_value(i,j) = (const2/2) * Q^(-2.1/1.1)
                     - const3 * point(i) * Q^(-3.2/1.1);
            end
```

```
        end
    end
end

if example == 12
    return_value = [3 1; 2 5];
end
```

```
function return_value = get_subgradient(example, point)
return_value = 0;
if example == 1
    return_value = 2*point;
end

if example == 2
    return_value = 2*point;
end

if example == 3
    return_value = 2*point;
end

if example == 4
    return_value = 2*point;
end

if example == 5
    return_value = 2*point;
end

if example == 6
    return_value = 2*point;
end

if example == 7
    return_value = 2*point;
end

if example == 389
    A = [100    100 10  5   10  0   0   25  0   10  55  5   45  20  0;
         90 100 10  35  20  5   0   35  55  25  20  0   40  25  10;
         70 50  0   55  25  100 40  50  0   30  60  10  30  0   40];
    b = [385; 470; 560];
    value = 0;
```

```
    for j = 1:3
        value(j) = 0;
        for i = 1:15
            value(j) = value(j) + A(j,i)*point(i)^2;
        end
        value(j) = value(j) - b(j);
    end
    [mini pos] = min(value);
    for i = 1:15
            return_value(i,1) = 2*A(pos,i)*point(i);
    end
end

if example == 389.1
    A = [100    100 10  5   10  0   0   25  0   10  55  5   45  20  0;
         90 100 10  35  20  5   0   35  55  25  20  0   40  25  10;
         70 50  0   55  25  100 40  50  0   30  60  10  30  0   40];
    b = [385; 470; 560];
    value = 0;
    for j = 1:3
        value(j) = 0;
        for i = 1:15
            value(j) = value(j) + A(j,i)*point(i)^2;
        end
        value(j) = value(j) - b(j);
    end
    [mini pos] = min(value);
    for i = 1:15
            return_value(i,1) = 2*A(pos,i)*point(i);
    end
end

if example == 10
    return_value = [2*point(1); 0.8*point(2); 1.2*point(3)];
end
```

```
if example == 12
    return_value = 2*point;
end
```

Bibliography

Censor, Y. and Gibali, A. (2008). Projections onto super-half-spaces for monotone variational inequality problems in finite-dimensional space. *Journal of Nonlinear and Convex Analysis*, 9:461–475.

Chan, D. and Pang, J.-S. (1982). Iterative methods for variational and complementarity problems. *Mathematical Programming*, 24:284–313.

Cruz, J. Y. B. and Iusem, A. N. (2010). Convergence of direct methods for paramonotone variational inequalities. *Applied Mathematics and Optimization*, 46:247 – 263.

Dafermos, S. (1980). Traffic equilibrium and variation inequalities. *Transportation Science*, 14:42–54.

Facchinei, F. and Pang, J.-S. (2003). *Finite-Dimensional Variational Inequalities and Complementarity Problems*. Springer Series in Operations Research. Springer-Verlag, New York. Published in two volumes, paginated continuously.

Fukushima, M. (1983). An outer appxoximation algorithm for solving general convex programs. *Operations Research*, 31(1):101–113.

Fukushima, M. (1986). relaxed projection method for variational inequalities. *Mathematical Programming*, 35:58–70.

Harker, P. (1984). A variational inequality approch for the determination of oligopolistic market equilibrium. *Mathematical Programming*, 30:105–111.

Harker, P. T. and Pang, J.-S. (1990). Finite dimensional variational inequality and non linear complementarity problems: A survey of theory, algorithms and applications. *Mathematical Programming*, 48:161–220.

He, B. (1997). A class of projection and contraction methods for monotone variational inequalities. *Applied Mathematics Optimization*, 35:69–76.

Josephy, N. (1979a). Quasi-newton methods for generalized equations.

Josephy, N. H. (1979b). Newton's method for generalized equations. Technical Summary Report 1965, Mathematics Research Center, University of Wisconsin-Madison, Madison, WI, USA. NTIS Accession No. AD A077 096.

Lawphongpanich, S. and Hearn, D. (1984). Simplicial decomposition of asymmetric traffic assignment problem. *Transportation Research*, 18B:123–133.

Ortega, J. M. and Rheinboldt, W. C. (1970). *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, New York.

Robinson, S. M. (1980). Strongly regular generalized equations. *Mathematics of Operations Research*, 5:43–62.

Robinson, S. M. (1991). An implicit-function theorem for a class of nonsmooth functions. *Mathematics of Operations Research*, 16:292–309.

Robinson, S. M. (1995). Sensitivity analysis of variational inequalities by normal-map techniques. In Giannessi, F. and Maugeri, A., editors, *Variational Inequalities and Network Equilibrium Problems*, pages 257–269, New York. Plenum Press.

Rockafellar, R. (1976). Augmented lagrangian and application of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1:97–116.

Rosen, J. and Suzuki, S. (1965). Construction of non-linear programming test problems. *Communications of The ACM*, 8(2):113.

Taji, K. and Fukushima, M. (1996). A new merit function and a successive quadratic programming algorithm for variational inequality problems. *SIAM Journal on Optimization*, 6(3):704–713.

Zhu, D. L. and Marcotte, P. (1994). An extended descent framework for variational inequalities. *Journal of optimization theory and applications*, 80(2):349–366.