A preconditioner for solving large scale variational inequality problems by a semismooth inexact approach

VALERIA RUGGIERO* and FEDERICA TINTI†
Dipartimento di Matematica, Università di Ferrara, Italy
(Received 00 Month 200x; In final form 00 Month 200x)

The numerical solution of a large scale variational inequality problem can be obtained using the generalisation of an Inexact Newton method applied to a semismooth nonlinear system. This approach requires a sparse and large linear system to be solved at each step. In this work we obtain an approximate solution of this system by the LSQR algorithm of Paige and Saunders combined with a convenient preconditioner that is a variant of the incomplete LU-factorization. Since the computation of the factorization of the preconditioning matrix can be very expensive and memory consuming, we propose a preconditioner that admits block-factorization. Thus the direct factorization is only applied to submatrices of smaller sizes. Numerical experiments on a set of test-problems arising from the literature show the effectiveness of this approach.

Keywords: Large scale variational inequality problems; semismooth nonlinear system; Inexact Newton method; LSQR method; incomplete LU preconditioner.

2000 Mathematics Subject Classifications: 65K10; 65Y20; 90C06

1 Introduction

We consider the classical variational inequality problem, denoted by VIP(F,C), which is to find \( x^* \in C \), such that

\[
<F(x^*), x - x^*> \geq 0, \forall x \in C
\]

where \( C \) is a nonempty closed convex subset of \( \mathbb{R}^n \), \( \langle \cdot, \cdot \rangle \) the usual inner product in \( \mathbb{R}^n \) and \( F : \mathbb{R}^n \rightarrow \mathbb{R}^n \) is a continuous function.

We assume that the feasible set \( C \) can be represented as follows

\[
C = \{ x \in \mathbb{R}^n | h(x) = 0, g(x) \geq 0, \Pi_l x \geq l, \Pi_u x \leq u \},
\]

where \( h(x) \), \( g(x) \), \( \Pi_l x \) and \( \Pi_u x \) denote affine constraints, lower and upper bounds, respectively.

---

Editorial note: This research was supported by the Italian Ministry for Education, University and Research (MIUR), FIRB Project RBAU01JYPN, Parallel Algorithms and Numerical Nonlinear Optimization.
where \( h : \mathbb{R}^n \rightarrow \mathbb{R}^p, g : \mathbb{R}^n \rightarrow \mathbb{R}^m \), \( \Pi_l \in \mathbb{R}^{nl \times n} \) and \( \Pi_u \in \mathbb{R}^{nu \times n} \), 
\( l \in \mathbb{R}^{nl}, u \in \mathbb{R}^{nu} \); \( \Pi_l \) (or \( \Pi_u \)) denotes the matrix given by the rows of the
identity matrix whose indices are equal to those of the entries of \( x \) which
are bounded below (above); \( nl \) and \( nu \) denote the number of entries of the
vector \( x \) subject to lower and upper bounds, respectively. If \( x_i \geq l_j \) for some
\( i \) and \( x_i \leq u_k \), we assume \( l_j \neq u_k \). This hypothesis means that there are
no fixed variables; otherwise, the problem can be reduced by eliminating them.

When problem (1) is a large scale variational inequality, a well known
approach is to find a numerical solution of the Karush–Kuhn–Tucker (KKT)
optimality conditions for VIP\((C,F)\):

\[
L(x, \lambda, \mu, \kappa_l, \kappa_u) = 0 \\
h(x) = 0 \\
\mu^T g(x) = 0 \quad g(x) \geq 0 \quad \mu \geq 0 \\
\kappa_l^T (\Pi_l x - l) = 0 \quad \Pi_l x - l \geq 0 \quad \kappa_l \geq 0 \\
\kappa_u^T (u - \Pi_u x) = 0 \quad u - \Pi_u x \geq 0 \quad \kappa_u \geq 0
\]

where \( L(x, \lambda, \mu, \kappa_l, \kappa_u) = F(x) - \nabla h(x) \lambda - \nabla g(x) \mu - \Pi_l^T \kappa_l + \Pi_u^T \kappa_u \) is the
Lagrangian function of the variational inequality. Here \( \nabla h(x)^T \) and \( \nabla g(x)^T \)
denote the Jacobian matrices of \( h(x) \) and \( g(x) \) respectively. A vector \( w = (x, \lambda, \mu, \kappa_l, \kappa_u) \in \mathbb{R}^{n+p+m+nl+nu} \) satisfying (3) is called a KKT-vector, while
the corresponding \( x \) vector is KKT-point. Subsequently, we will often refer to
a KKT-vector simply as a KKT-point.

For the relationship between a KKT point and the solution of VIP\((F,C)\)
see [12].

In [6] (see also [7]), from Fischer and Burmeister’s function [9], the KKT-
conditions for variational inequality problems are reformulated as a system
of nonlinear semismooth equations \( \Phi(w) = 0 \), that can be solved by a non–
smooth Newton method [20]. A similar approach has recently been proposed
in [3], where the system of nonlinear semismooth equations is solved by a
quasi–Newton method for non–smooth equations. In both cases, a local algo-
ритm can be devised for locating a KKT-vector for VIP\((C,F)\). Furthermore,
in [6] the local algorithm is globalized by damping the search direction to
force a sufficient decrease of a merit function, such as the least square merit
function \( \Psi(w) = \frac{1}{2} \| \Phi(w) \|^2 \). Here \( \| \cdot \| \) indicates the Euclidean norm. Thus,
at each step of this iterative scheme, the computation of the solution of a
sparse linear system is required. In the case of large scale problems, the use
of direct solvers can be very expensive and memory consuming. In [7], the
nonlinear semismooth system is solved by the generalisation of the Inexact
Newton method (see [13]), that enables us to use an iterative solver for the inner linear systems. Following this approach, even if the descent direction for the merit function is computed approximately at each step, the local and global convergence of the whole method is still guaranteed. Nevertheless, this theoretical result is not exploited in the state-of-the-art software [16]. Indeed, also when an iterative solver is used, the linear inner system is solved at the machine precision.

On the contrary, to take full advantage of this approach, we can use for the inner scheme an adaptive stopping rule that allows us to avoid unnecessary inner iterations when the current iterate is far from the solution, (i.e. for the initial outer iterations).

In this paper, we devise a suitable adaptive rule that can be included in the software and that assures us to determine a descent direction for the merit function. This procedure, combined by a backtracking rule, enables us to obtain a method globally convergent.

The numerical experiments reported on the last section are carried out by a code implementing this rule; the results show that the inexact solution of the inner systems does not affect the global convergence.

Furthermore, it is well known previously, (see [16]), that an effective iterative inner solver is the LSQR method [18] combined with a suitable preconditioner, for example the incomplete $LU$ factorization. Nevertheless, for large scale problems, the computation of this sparse preconditioner can still be expensive and memory consuming as arises for the direct solvers. For example, this happens when all the variables $x$ have to satisfy box constraints. In order to avoid these drawbacks, we introduce a preconditioner that admits block-factorization; in this way, we compute an incomplete $LU$ factorization by factorizing submatrices of smaller sizes.

This preconditioner enables us to obtain better performance of the semismooth inexact method in terms of execution time and total number of inner iterations.

The paper is organised as follows. In the next section, we describe the semismooth inexact method.

In section 3 we discuss the solution of the inner linear system using the LSQR method combined with the proposed preconditioner.

Finally in order to evaluate the effectiveness of the proposed approach, we report the numerical results obtained on some well-known test problems.

For the sake of completeness, here we report some basic definitions.

**Definition 1.1** (see [19]) Let $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a locally Lipschitzian near a given point $x$, where $x \in \Omega_F$, with $\Omega_F$ the set of points at which $F$ fails to be
differentiable. The $B$-subdifferential of $F$ at $x$ is

$$\partial_B F(x) = \{ Z \in \mathbb{R}^{m \times n} \text{ s.t. } \exists \{ x^k \} \not\in \Omega_F, \text{ with } \lim_{x^k \to x} \nabla F(x^k)^T = Z \}.$$ 

Here $\nabla F(x^k)^T$ denotes the Jacobian matrix of $F$ at $x^k$.

In [2], the convex hull of the $B$-subdifferential:

$$\partial F(x) = \text{co} \partial_B F(x),$$

is referred as Clarke’s generalized Jacobian.

**Definition 1.2** (see [20]) Let $F: \mathbb{R}^n \to \mathbb{R}^m$ be locally Lipschitz function at $x \in \mathbb{R}^n$. We say that $F$ is semismooth at $x$ if

$$\lim_{Z \in \partial F(x+tv^t) \atop v^t \to v, t \downarrow 0} Z v^t$$

exists for all $v \in \mathbb{R}^n$.

**Definition 1.3** (see [8]) Suppose that $F$ is semismooth at $x$. We say that $F$ is strongly semismooth at $x$ if for any $Z \in \partial F(x+v)$ and for any $v \to 0$,

$$Zv - F'(x; v) = O(\|v\|^2),$$

where $F'(x; v)$ is the classical directional derivative.

## 2 The semismooth inexact method

In this section, we describe the semismooth inexact method. The zeros of the two-variable convex function $\varphi: \mathbb{R}^2 \to \mathbb{R}$, introduced in [9],

$$\varphi(a, b) := \sqrt{a^2 + b^2} - a - b$$

satisfy the following property

$$\varphi(a, b) = 0 \iff a \geq 0, b \geq 0, ab = 0.$$ 

(4)

Note that $\varphi$ is continuously differentiable everywhere but in the origin.
If \( a \in \mathbb{R}^n, b \in \mathbb{R}^n \), we can define
\[
\phi(a, b) = \begin{pmatrix}
\varphi(a_1, b_1) \\
\varphi(a_2, b_2) \\
\vdots \\
\varphi(a_n, b_n)
\end{pmatrix}.
\]

The property (4) holds for each component of \( \phi(a, b) \).
Thus, we have:
- if \(|a_i| + |b_i| \neq 0\) for \( i = 1, \ldots, n \), then \( \phi(a, b) \) is a differentiable function in \((a, b)\);
- if there exists an index \( i \) such that \(|a_i| + |b_i| = 0\), then \( \phi(a, b) \) is a semismooth function in \((a, b)\).

The gradient of each component of \( \phi \) is
\[
\nabla \varphi(a_i, b_i) = \begin{bmatrix}
\frac{a_i}{\sqrt{a_i^2 + b_i^2}} - 1 \\
\frac{b_i}{\sqrt{a_i^2 + b_i^2}} - 1
\end{bmatrix}.
\]

If \((a_i, b_i) = (0, 0)\), we can define the **Clarke’s generalised gradient** as
\[
\partial \varphi(0, 0) = \text{co} \bigcup_{\phi \in [0, 2\pi]} \begin{bmatrix}
\cos \phi - 1 \\
\sin \phi - 1
\end{bmatrix}.
\]

Then the point \((-1, -1)^T\) belongs to \(\partial \varphi(0, 0)\).

Considering the property (4), the KKT-conditions (3) can be equivalently written as the following system of nonlinear equations
\[
F(x) - \nabla h(x)\lambda - \nabla g(x)\mu - \Pi_l^T \kappa_l + \Pi_u^T \kappa_u = 0
\]
\[
h(x) = 0
\]
\[
\varphi_l(\mu, g(x)) = 0
\]
\[
\varphi_l(\kappa_l, \Pi_l x - l) = 0
\]
\[
\varphi_u(\kappa_u, u - \Pi_u x) = 0
\]
or, in a more concise form,
\[
\Phi(w) = 0,
\]
(5)
A preconditioner for solving large scale VIPs by a semismooth inexact approach

where \( w = (x^T, \lambda^T, \mu^T, \kappa_l^T, \kappa_u^T) \in \mathbb{R}^{n+p+m+n_l+n_u}, \varphi_I : \mathbb{R}^{2m} \rightarrow \mathbb{R}^m \) with \( \varphi_I(\mu, g(x)) := (\varphi(\mu_1, g_1), \ldots, \varphi(\mu_m, g_m)) \in \mathbb{R}^m \), \( \varphi_l : \mathbb{R}^{2n_l} \rightarrow \mathbb{R}^{n_l} \), with \( \varphi_l(\kappa_l, l - \Pi_l x) \in \mathbb{R}^{n_l} \), \( \varphi_u : \mathbb{R}^{2n_u} \rightarrow \mathbb{R}^{n_u} \), with \( \varphi_u(\kappa_u, u - \Pi_u x) \in \mathbb{R}^{n_u} \).

Since \( \varphi_I, \varphi_l, \varphi_u \) are semismooth functions, then the system (5) is a non–smooth reformulation of the KKT-conditions (3).

Indeed, the function \( \Phi \) is a locally Lipschitz-continuous operator and the following properties hold [6].

**Proposition 2.1**

a) Assume that \( F \) is semismooth and that all the components of the functions \( g \) and \( h \) are continuously differentiable with semismooth gradients. Then \( \Phi \) is semismooth.

b) Assume that \( F \) is strongly semismooth and that all the components of the functions \( g \) and \( h \) are continuously differentiable with strongly semismooth gradients. Then \( \Phi \) is strongly semismooth.

The proof of this proposition reported in [6] exploits the fact that the composite of (strongly) semismooth functions is (strongly) semismooth (see [15] and [10]).

The system (5) can be solved by the non–smooth Inexact Newton method, ([7], [13]), given by

\[
w^{k+1} = w^k + \alpha_k \Delta w^k,
\]

where \( w^0 \) is a starting point, \( \alpha_k \) is a damping parameter (determined by a line search so that the Armijo rule is satisfied) and \( \Delta w^k \) is the solution of the following linear system

\[
H_k \Delta w = -\Phi(w^k) + r^k.
\] (6)

Here \( H_k \in \partial \Phi(w^k) \), \( r^k \) satisfies the residual condition, \( \| r^k \| \leq \eta_k \| \Phi(w^k) \| \), where \( \eta_k \leq \bar{\eta} < 1 \) is the forcing term.

By permuting the block equations of the system (6) and changing the sign of the fourth block equation, we can write (we omit the index of the step iteration \( k \)):

\[
\begin{bmatrix}
D_{\kappa_l} & 0 & 0 & D_I \Pi_I & 0 \\
0 & D_{\kappa_u} & 0 & D_u \Pi_u & 0 \\
0 & 0 & D_\mu & D_g(\nabla g(x))^T & 0 \\
\Pi_I^T & -\Pi_u^T & \nabla g(x) & \nabla z L(x) & \nabla h(x) \\
0 & 0 & 0 & (\nabla h(x))^T & 0
\end{bmatrix}
\begin{bmatrix}
\Delta \kappa_l \\
\Delta \kappa_u \\
\Delta \mu \\
\Delta x \\
\Delta \lambda
\end{bmatrix}
= \begin{bmatrix}
D_{\kappa_l} & 0 & 0 & D_I \Pi_I & 0 \\
0 & D_{\kappa_u} & 0 & D_u \Pi_u & 0 \\
0 & 0 & D_\mu & D_g(\nabla g(x))^T & 0 \\
\Pi_I^T & -\Pi_u^T & \nabla g(x) & \nabla z L(x) & \nabla h(x) \\
0 & 0 & 0 & (\nabla h(x))^T & 0
\end{bmatrix}
\begin{bmatrix}
\Delta \kappa_l \\
\Delta \kappa_u \\
\Delta \mu \\
\Delta x \\
\Delta \lambda
\end{bmatrix}
\]
A preconditioner for solving large scale VIPs by a semismooth inexact approach

\[
\begin{align*}
    \varphi_l(\kappa_l, \Pi_l x - l) \\
    \varphi_u(\kappa_u, u - \Pi_u x) \\
    \varphi_I(\mu, g(x)) \\
    h(x)
\end{align*}
\]

\[= - \begin{bmatrix}
    \varphi_l(\kappa_l, \Pi_l x - l) \\
    \varphi_u(\kappa_u, u - \Pi_u x) \\
    \varphi_I(\mu, g(x)) \\
    h(x)
\end{bmatrix} + Pr
\]

where \( \nabla_x L(w) = -\nabla F(x) + \sum_{i=1}^m \nabla^2 g_i(x) \mu_i + \sum_{i=1}^p \nabla^2 h_i(x) \lambda_i \); \( \nabla^2 g_i(x) \) and \( \nabla^2 h_i(x) \) are Hessian matrices of inequality and equality constraints; \( r \) is the residual vector and \( P \) is the permutation matrix, and

\[-\alpha = F(x) - \nabla h(x) \lambda - \nabla g(x) \mu - (\Pi_l)^T \kappa_l + (\Pi_u)^T \kappa_u.\]

Furthermore, we have

\[D_g = \text{diag}(d_{g_1}, \ldots, d_{g_m})\]

\[(d_{g_i}) = \begin{cases} 
    \left( \frac{g_i}{\sqrt{\mu_i^2 + g_i^2}} \right) - 1 & \text{if } (g_i(x), \mu_i) \neq 0 \\
    -1 & \text{if } (g_i(x), \mu_i) = 0
\end{cases}\]

\[D_\mu = \text{diag}(d_{\mu_1}, \ldots, d_{\mu_m})\]

\[(d_{\mu_i}) = \begin{cases} 
    \left( \frac{\mu_i}{\sqrt{\mu_i^2 + g_i^2}} \right) - 1 & \text{if } (g_i(x), \mu_i) \neq 0 \\
    -1 & \text{if } (g_i(x), \mu_i) = 0
\end{cases}\]

\[D_l = \text{diag}(d_{l_1}, \ldots, d_{l_n})\]

\[(d_{l_i}) = \begin{cases} 
    \left( \frac{(\Pi_l x)_i - l_i}{\sqrt{\kappa_l^2 + (\Pi_l x)_i - l_i)^2}} \right) - 1 & \text{if } ((\Pi_l x)_i - l_i, \kappa_l) \neq 0 \\
    -1 & \text{if } ((\Pi_l x)_i - l_i, \kappa_l) = 0
\end{cases}\]

\[D_{\kappa_l} = \text{diag}(d_{\kappa_{l_1}}, \ldots, d_{\kappa_{l_n}})\]

\[(d_{\kappa_{l_i}}) = \begin{cases} 
    \left( \frac{\kappa_{l_i}}{\sqrt{\kappa_{l_i}^2 + ((\Pi_l x)_i - l_i)^2}} \right) - 1 & \text{if } (\kappa_{l_i}, (\Pi_l x)_i - l_i) \neq 0 \\
    -1 & \text{if } (\kappa_{l_i}, (\Pi_l x)_i - l_i) = 0
\end{cases}\]
A preconditioner for solving large scale VIPs by a semismooth inexact approach

\[ D_u = \text{diag}(d_{u_1}, \ldots, d_{u_n}) \]

\[
(d_{u_i}) = \begin{cases} \left( \frac{(u_i - (\Pi u)x)_i}{\sqrt{\kappa_{u i}^2 + (u_i - (\Pi u)x)_i^2}} - 1 \right) i f (u_i - (\Pi u)x)_i, \kappa_{u i}) \neq 0 \\
-1 \text{ if } (u_i - (\Pi u)x)_i, \kappa_{u i}) = 0 
\end{cases}
\]

\[ D_{\kappa u} = \text{diag}(d_{\kappa u_1}, \ldots, d_{\kappa u_n}) \]

\[
(d_{\kappa u_i}) = \begin{cases} \left( \frac{\kappa_{u i}}{\sqrt{\kappa_{u i}^2 + (u_i - (\Pi u)x)_i^2}} - 1 \right) i f (\kappa_{u i}, u_i - (\Pi u)x)_i) \neq 0 \\
-1 \text{ if } (\kappa_{u i}, u_i - (\Pi u)x)_i = 0 
\end{cases}
\]

In [7], Theorem 4.3 summarises the main convergence properties of the non-smooth Inexact Newton method for VIP(C,F).

For completeness, we report the cited theorem.

**Theorem 2.2** (see Theorem 4.3 in [7]) Let \( F \) be a \( C^1 \)-function and \( g \) and \( h \) be \( C^2 \)-functions. Assume that \( \| r^k \| \leq \eta_k \| \Phi(w^k) \| \), where \( \eta_k \) is a sequence of positive numbers such that \( \eta_k \leq \eta \) for every \( k \) with an arbitrary \( \eta \in (0, 1) \).

Then the following assertions hold:

(a) each accumulation point of the sequence \( \{w^k\} \) generated by the algorithm is a stationary point of \( \Psi = \frac{1}{2}\|\Phi(w)\|^2 \);

(b) if the sequence \( \{w^k\} \) has an accumulation point, say \( w^* \), which is an isolated KKT–point of VIP(C,F), then \( w^k \) converges to \( w^* \);

(c) if the sequence \( \{w^k\} \) has an accumulation point, say \( w^* \), which is a solution of the system \( \Phi(w^*) = 0 \) and where \( \partial_B \Phi(w^*) \) exists and all elements in the B–subdifferential \( \partial_B \Phi(w^*) \) are nonsingular, then \( \{w^k\} \) converges to \( w^* \). Moreover, if \( \{\eta_k\} \) tends to 0, then \( \{w^k\} \) converges to \( w^* \) Q–superlinearly. Furthermore if \( \eta_k = O(\|\Phi(w^k)\|) \), if \( F \) is an \( LC^1 \)–function\(^1\) and if \( g \) and \( h \) are \( LC^2 \)–functions\(^1\), then the convergence rate is Q–quadratic.

The semismooth inexact method for the solution of the system (5), can be expressed as follows:

**Step 1** Set: \( w^0 = (x^0, \lambda^0, \mu^0, \kappa^0, \kappa^0_l, \kappa^0_u) \in \mathbb{R}^{n+p+m+n+l+nu}, \rho > 0, \beta \in (0, 1), \sigma \in (0, 1), q > 2; \lambda^0 = 0, \mu^0 = 0, \kappa^0 = 0, \kappa^0_l = 0, \kappa^0_u = 0, \eta^0 = 0; \)

**Step 2** Define a stopping criterion.

\(^1\) A \( C^k \)–function is called \( LC^k \)–function if its \( k \)–th derivative is locally Lipschitz-continuous.
Step 3 Select an element $H_k \in \partial_B \Phi(w^k)$ and compute the direction $\Delta w$:

**find the approximate solution** $\Delta w^k \in \mathbb{R}^n$ of the system

$$H_k \Delta w^k = -\Phi(w^k) + r^k$$

such that $\|r^k\| \leq \eta_k \|\Phi(w^k)\|$. 

If this is not possible or if the condition

$$\nabla \Psi(w^k)^T \Delta w^k \leq -\rho \|\Delta w^k\|^q$$

is not satisfied, then set $\Delta w^k := -\nabla \Psi(w^k) = -(H_k)^T \Phi(w^k)$.

Step 4 Select a linesearch strategy:

compute the minimum integer $l$, such that, if $\alpha_k = \beta^l$ the following condition holds

$$\Psi(w^k + \alpha_k \Delta w^k) \leq \Psi(w^k) + \sigma \alpha_k \nabla \Psi(w^k)^T \Delta w^k.$$  

Step 5 Compute $w^{k+1} = w^k + \alpha_k \Delta w^k$, $k = k + 1$, choose $\eta_k \geq 0$ and go to **Step 2**.

In step 2, the algorithm stops when one of these conditions holds:

(i) the number of iterations is larger than a fixed-value, called \textit{maxiter};

(ii) $\|\Phi(w^k)\| \leq \tau_1$;

(iii) $\left|\|\Phi(w^k)\| - \|\Phi(w^{k-1})\|\right| < \tau_2$.

where $\tau_1$ and $\tau_2$ are prefixed tolerances.

3 **Iterative method for solving inner linear systems.**

In order to solve the Newton equation $H_k \Delta w^k = -\Phi(w^k)$, we can use the iterative LSQR method [18]. A similar approach has also been used in [16]. The LSQR method is a stable version of the conjugate gradient algorithm applied to the normal equations

$$H_k^T H_k \Delta w = -H_k^T \Phi(w^k).$$

In practice, the LSQR iterative scheme determines an approximate solution of the least squares problem

$$\min_{\Delta w} \|H_k \Delta w + \Phi(w^k)\|^2$$
and then the resulting $\Delta w$ is used as a descent direction for the merit function $\Psi(w)$. Indeed, the following proposition can be proved.

**Proposition 3.1** Let $\Delta w_i$ be the $i$-th iteration of the LSQR scheme applied to the least squares problem (11). If $\Delta w_i$ is such that

$$\|H_k \Delta w_i + \Phi(w^k)\| \leq \eta_k \|\Phi(w^k)\|,$$

then $\Delta w_i$ is a descent direction for the merit function $\Psi(w)$ at $w^k$.

**Proof** If $t_i = -(H_k \Delta w_i + \Phi(w^k))$ is the residual of the $i$-th iteration of the inner solver and the condition (12) is satisfied, then we have

$$\nabla \Psi(w^k)^T \Delta w_i = \Phi(w^k)^T H_k \Delta w_i$$

$$= \Phi(w^k)^T (-\Phi(w^k) - t_i)$$

$$\leq -\|\Phi(w^k)\|^2 + \|\Phi(w^k)\| \cdot \| - t_i\|$$

$$\leq -(1 - \eta_k)\|\Phi(w^k)\|^2 \leq 0$$

Since $\eta_k \leq \eta < 1$, $\nabla \Psi(w^k)^T \Delta w_i < 0$ when $\Phi(w^k) \neq 0$. Then we put $\Delta w_i = \Delta w^k$ and $-t_i = r^k$. □

The LSQR iterative method can also be applied when $H_k$ is a singular matrix, and this arises in many real applications. The implementation of LSQR uses the termination rules developed by Paige and Saunders in [18].

(i) $\text{cond}(H_k) \geq \text{CONLIM}$

(ii) $\|t_i\| \leq BTOL \|\Phi(w^k)\| + ATOL \|H_k\| \|\Delta w_i\|$ 

(iii) $\frac{\|H_k^T t_i\|}{\|H_k\| \|t_i\|} \leq \text{ATOL}$

where $\|H_k\|$ and $\text{cond}(H_k)$ are estimates of the spectral norm of the matrix $H_k$ and the condition number of $H_k$ computed by LSQR routine.

Standard choices for the parameters $\text{ATOL}$, $\text{BTOL}$ and $\text{CONLIM}$ used in [16] are:

$$\text{ATOL} = \text{BTOL} = \epsilon^\frac{2}{3}, \quad \text{CONLIM} = \frac{1}{10 \sqrt{\epsilon}},$$

where $\epsilon$ is the machine precision.

These termination rules determine a numerical approximation to the solution of the least squares problems close to the exact solution as soon as possible.
A preconditioner for solving large scale VIPs by a semismooth inexact approach

with respect to the machine arithmetic.

In this paper, we propose to exploit an adaptive termination rule as stated by the Theorem 2.2 and 3.1. This choice enables us to reduce the number of inner iterations when the current iterate is far from the solution. In particular, we propose to use the following adaptive termination rule:

\[ ATOL = 0, \quad BTOL = \eta_k \]  \hspace{1cm} (14)

with

\[ \eta_k = \max \left\{ \min \left\{ \frac{1}{1+k}, \min \left\{ 0.1, \frac{10}{\| \Psi(w^0) \|} \right\} \right\}, 10^{-8} \right\}, \]  \hspace{1cm} (15)

so that \( \eta_k \leq \frac{1}{2} \). This choice modifies the total number of inner iterations of the semismooth algorithm.

The code implementing the LSQR algorithm requires a workspace of few vectors of size \( H_k \) elements, with size \( H_k = n + p + m + nl + nu \), (order of \( H_k \)). In addition, \( \Delta w, \Phi(w^k) \) and the nonzero elements \( nelem \) of \( H_k \) have to be stored. This last matrix is stored in compressed storage format [21].

The cost per iteration consists of three matrix-vector products \( H_k x \) or \( H_k^T y \), where the sparsity of the matrix is exploited.

In order to accelerate the rate of convergence of the LSQR method, we can use a right preconditioner for the system (10).

This means that, instead of solving (10), we solve:

\[ \min \| H_k M^{-1} \Delta z + \Phi(w^k) \|_2, \]  \hspace{1cm} (16)

with \( \Delta z = M \Delta w \) (where \( M \) is a nonsingular matrix), or

\[ M^{-T} H_k^T H_k M^{-1} \Delta z = -M^{-T} H_k^T \Phi(w^k) \]  \hspace{1cm} (17)

A standard choice for \( M \) is the incomplete LU factorization of \( H_k \). This preconditioner can be obtained by the sparse factorization routine lu1fac of the MINOS package (see [11], [17]). This routine computes a factorization of \( H_k \) with a Threshold Partial Pivoting (TPP) technique, where an absolute tolerance is chosen for treating reals as zeros; dense columns are treated later.

In this case, for any outer iteration of the semismooth method, we have to compute the incomplete LU factorization of \( H_k \). Furthermore, at each iteration of the LSQR scheme we have to solve the system \( LUx = y \) and \( U^T L^T x = z \); additional workspace for some temporary vectors can be reserved, but the main request of memory is due to the storage of the nonzero elements of the sparse factors \( L \) and \( U \).
In the first part of Table 1, we report, for some test problems described in Tables 2 and 3, the time expressed in seconds to compute the incomplete LU factorization at the first iteration (denoted by “time(ILU(H_k))”), the total time of the first iteration\(^2\) in seconds (denoted by “time1”), and the numbers of the nonzero elements of H\(_k\), L and U (denoted by nelem(H\(_k\)), LenL and LenU respectively).

We observe in the first part of the Table 1 that the storage required by the L and U goes up to three times that required by H\(_k\) and the factorization time requires a great part of the time of one iteration. Thus the task of determining the incomplete LU factorization of H\(_k\) is very expensive in terms of time and memory consuming and we loose the advantages to use an iterative approach instead of a direct approach.

In order to avoid this drawback, we can modify the preconditioner so that we have to apply the incomplete LU factorization only to a submatrix of H\(_k\).

The approach consists in to approximate the two diagonal matrices D\(_{ki}\) and D\(_{ku}\) by the diagonal matrices \(\tilde{D}_{ki}\) and \(\tilde{D}_{ku}\), defined as follows: whenever the absolute values of the diagonal entries of the matrices D\(_{ki}\) and D\(_{ku}\) are too small, we set the corresponding diagonal entries \(\tilde{d}_{ki}\) and \(\tilde{d}_{ku}\) of \(\tilde{D}_{ki}\) and \(\tilde{D}_{ku}\) equal to \(-\tau\), that is

\[
\text{if } |d_{ki}| \leq \tilde{\epsilon} \text{ then } \tilde{d}_{ki} = -\tau \text{ else } \tilde{d}_{ki} = d_{ki},
\]

\[
\text{if } |d_{ku}| \leq \tilde{\epsilon} \text{ then } \tilde{d}_{ku} = -\tau \text{ else } \tilde{d}_{ku} = d_{ku},
\]

where \(\tilde{\epsilon}\) is related to the machine precision and \(\tau = -\tilde{\epsilon}^3\).

When, for some \(i\), \((\Pi_l x)_i = l_i\) (or \((\Pi_u x)_i = u_i\)), we observe that \(d_{ki} = 0\) (or \(d_{ku} = 0\)).

Thus we consider the matrix \(\hat{H} = Lu\) where

\[
L = \begin{bmatrix}
I & 0 & 0 & 0 & 0 \\
0 & I & 0 & 0 & 0 \\
0 & 0 & \Pi_l^T(\tilde{D}_{ki})^{-1} - \Pi_l^T(\tilde{D}_{ku})^{-1} & \tilde{L}
\end{bmatrix}
\]

\(^2\)In this case the solution of the inner system is computed at the machine precision.
\[
\mathcal{U} = \begin{bmatrix}
\tilde{D}_{\kappa_l} & 0 & 0 & D_l \Pi_l & 0 \\
0 & \tilde{D}_{\kappa_u} & 0 & D_u \Pi_u & 0 \\
0 & 0 & 0 & \hat{U} \\
0 & 0 & 0 & 0 & \hat{U}
\end{bmatrix}
\]

and \(\hat{L}\) and \(\hat{U}\) are the factors of the incomplete factorization of the matrix \(\hat{B}\):

\[
\hat{B} = \begin{bmatrix}
D_{\mu} & D_{g} (\nabla g(x))^T & 0 \\
\nabla g(x) & A & \nabla h(x) \\
0 & (\nabla h(x))^T & 0
\end{bmatrix}
\]

with \(A = \nabla_x L(w) - (\Pi_l)^T (\tilde{D}_{\kappa_l})^{-1} D_l \Pi_l + (\Pi_u)^T (\tilde{D}_{\kappa_u})^{-1} D_u \Pi_u\).

Using the matrix \(\hat{H} = \hat{L} \hat{U}\) as preconditioner in LSQR-method the problem (16) becomes

\[
\min ||H_k \mathcal{U}^{-1} \mathcal{L}^{-1} \Delta z + \Phi(w^k)||^2,
\]

with \(\Delta z = \hat{L} \hat{U} \Delta w\).

Instead of storing the nonzero elements of \(H\) and the incomplete factors \(L\) and \(U\), we store the submatrix \(\hat{B}\), the incomplete factors \(\hat{L}\) and \(\hat{U}\) and the nonzero blocks of the matrices \(\mathcal{L}\) and \(\mathcal{U}\).

The matrices \(\hat{L}\) and \(\hat{U}\) are obtained by applying the MINOS’s TPP factorization to the submatrix \(\hat{B}\) of \(\hat{H}\) instead of whole matrix \(\hat{H}\).

In Table 1, we compare the two incomplete factorizations described above. With respect to the second part of the table, we denote by \(\text{size}(\hat{B})\) the order of the matrix \(\hat{B}\) (i.e. \(\text{size}(\hat{B}) = n + \text{neq} + m\)), by \(\text{nelem}(\hat{B})\), \(\text{Len}\hat{L}\), \(\text{Len}\hat{U}\) the number of nonzero elements of the matrices \(\hat{B}\), \(\hat{L}\), \(\hat{U}\) respectively.

In this case the storage required by \(\hat{L}\) and \(\hat{U}\) decreases up to 13% - 15% of that required by \(L\) and \(U\), allowing the factorization of problems of larger size. In addition, we remark a reduction in time by using the incomplete factorization of the matrix \(\hat{B}\) instead of the incomplete factorization of the matrix \(H_k\).

Indeed, if we consider the value \(\bar{t} = \text{time1} - \text{time}(ILU(H_k)) + \text{time}(ILU(\hat{B}))\), we observe that \(\bar{t}\) is equal or even greater than the time “time2” of one iteration when we use the incomplete factorization of \(\hat{B}\).

Then the reduction of the time of the factorization produces a reduction time of the outer iteration of the method up to 25%.

In this way, if \(nl\) and \(nu\) are close to \(n\), then we have a considerable decrease in the required storage.
Table 1. Computational cost for incomplete LU factorization

<table>
<thead>
<tr>
<th>Test Problem</th>
<th>size($H_k$)</th>
<th>nelem($H_k$)</th>
<th>time($ILU(H_k)$)</th>
<th>time1 LenL</th>
<th>LenU</th>
</tr>
</thead>
<tbody>
<tr>
<td>mitt480</td>
<td>38400</td>
<td>146560</td>
<td>3.17</td>
<td>5.17</td>
<td>345386</td>
</tr>
<tr>
<td>mitt480</td>
<td>46600</td>
<td>185580</td>
<td>4.65</td>
<td>8.38</td>
<td>442699</td>
</tr>
<tr>
<td>mitt495</td>
<td>54150</td>
<td>206815</td>
<td>5</td>
<td>9.04</td>
<td>498312</td>
</tr>
<tr>
<td>Obstacle100</td>
<td>300000</td>
<td>109600</td>
<td>0.43</td>
<td>0.94</td>
<td>129147</td>
</tr>
<tr>
<td>Obstacle200</td>
<td>120000</td>
<td>439200</td>
<td>3.48</td>
<td>4.68</td>
<td>566624</td>
</tr>
<tr>
<td>Obstacle300</td>
<td>270000</td>
<td>988800</td>
<td>10.85</td>
<td>14.02</td>
<td>1325945</td>
</tr>
<tr>
<td>Obstacle400</td>
<td>480000</td>
<td>1758400</td>
<td>20.25</td>
<td>25.98</td>
<td>2421856</td>
</tr>
<tr>
<td>Obstacle500</td>
<td>750000</td>
<td>2748000</td>
<td>40.53</td>
<td>48.51</td>
<td>3887696</td>
</tr>
<tr>
<td>Obstacle600</td>
<td>1080000</td>
<td>3957600</td>
<td>65.52</td>
<td>89.63</td>
<td>5721100</td>
</tr>
<tr>
<td>Obstacle700</td>
<td>1470000</td>
<td>5387200</td>
<td>90.92</td>
<td>123.52</td>
<td>7897344</td>
</tr>
<tr>
<td>Obstacle800</td>
<td>1920000</td>
<td>7036800</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Problem</th>
<th>size($\hat{B}$)</th>
<th>nelem($\hat{B}$)</th>
<th>time($ILU(\hat{B})$)</th>
<th>time2 LenL</th>
<th>LenU</th>
</tr>
</thead>
<tbody>
<tr>
<td>mitt480</td>
<td>192000</td>
<td>88960</td>
<td>3.09</td>
<td>4.80</td>
<td>326186</td>
</tr>
<tr>
<td>mitt490</td>
<td>243000</td>
<td>112680</td>
<td>4.46</td>
<td>7.76</td>
<td>418399</td>
</tr>
<tr>
<td>mitt495</td>
<td>27075</td>
<td>125590</td>
<td>4.89</td>
<td>8.42</td>
<td>471237</td>
</tr>
<tr>
<td>Obstacle100</td>
<td>10000</td>
<td>49600</td>
<td>0.42</td>
<td>0.71</td>
<td>107711</td>
</tr>
<tr>
<td>Obstacle200</td>
<td>40000</td>
<td>199200</td>
<td>2.76</td>
<td>3.75</td>
<td>482530</td>
</tr>
<tr>
<td>Obstacle300</td>
<td>90000</td>
<td>448800</td>
<td>8.53</td>
<td>11.58</td>
<td>1133552</td>
</tr>
<tr>
<td>Obstacle400</td>
<td>160000</td>
<td>798400</td>
<td>16.65</td>
<td>22.18</td>
<td>2090328</td>
</tr>
<tr>
<td>Obstacle500</td>
<td>250000</td>
<td>1248000</td>
<td>30.18</td>
<td>40.10</td>
<td>3364029</td>
</tr>
<tr>
<td>Obstacle600</td>
<td>360000</td>
<td>1797600</td>
<td>51.85</td>
<td>82.06</td>
<td>4964128</td>
</tr>
<tr>
<td>Obstacle700</td>
<td>490000</td>
<td>2447200</td>
<td>78.72</td>
<td>161.18</td>
<td>6905158</td>
</tr>
<tr>
<td>Obstacle800</td>
<td>640000</td>
<td>3196800</td>
<td>118.22</td>
<td>270</td>
<td>9133646</td>
</tr>
<tr>
<td>Obstacle900</td>
<td>810000</td>
<td>4046400</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

*The $LU$ factorization cannot be carried out by MINOS.

4 Numerical Results

We have built two versions of the non–smooth Inexact Newton method combined with the LSQR scheme:

- in the first version, named $VsemiLSQR$, we use the incomplete $LU$ factorization of $H_k$ as preconditioner (computed by lu1fac of MINOS applied to the whole matrix $H_k$);
- in the second version, named $VsemiLSQRnew$, we use the incomplete $LU$ factorization of the submatrix $\hat{B}$ as preconditioner (computed by the lu1fac routine).

The two different versions of the non–smooth Inexact Newton method are carried out in two different ways:

- in the first case, $VsemiLSQR$ and $VsemiLSQRnew$ use $ATOL = BTOL = \epsilon^{2/3}$ as the inner termination rule for the LSQR scheme; then the new solution is computed at maximum precision with respect to the machine precision;
in the second case, the codes use an adaptive inner stopping rule for the LSQR scheme, the parameters ATOL and BTOL are as in (14) and (15); in this case the names of the codes have the suffix 2, i.e. \textit{VIsemiLSQR2} and \textit{VIsemiLSQRnew2}, respectively.

The matrices $H_k$ and $\hat{B}$ are stored as required by \texttt{lu1fac} routine. All the other sparse matrices are stored in a compressed sparse column format [21].

In order to evaluate the effectiveness of the two approaches, we have carried out the codes on a workstation DEC ALPHA 21264 EV6/7 633Mhz with 512MB RAM and Linux RH7.2alpha operating system. All codes have been compiled with the +O3 optimisation option of the ALPHA Fortran compiler.

In all codes, the input parameters are $\beta = 0.5$, $\sigma = 10^{-4}$ and we declare a failure of the algorithm (denoted by *) when one of the following situations happens:

- the tolerances $\tau_1 = 10^{-8}$, $\tau_2 = 10^{-9}$ are not reached after 500 iterations;
- in order to satisfy the backtracking condition (9), more than 30 reductions of the damping parameter have been performed;
- the incomplete \textit{LU} factorization did not obtain success (i.e. $\text{inform} \neq 0$ in \texttt{lu1fac} procedure).

Furthermore, when the incomplete \textit{LU} factorization is carried out, we do not check the descent condition (12) but always prefer to use the Newton direction. Indeed, numerical experiment shows that the choice of the step gradient produces a slower convergence rate or a failure of the method (i.e. number of external iterations greater than 500).

We have considered a test set composed of nonlinear programming problems (i.e. NLPs), mixed complementarity problems (i.e. MCPs) and small variational inequality problems (i.e. Small VIPs). They are listed in Tables 2 and 3, where we report the number of variables $n$, number of equality and inequality constraints, $p$ and $m$ respectively, number of the lower and upper bounds, $nl$ and $nu$ respectively, and references.

In Tables 4 and 5 we report the results obtained for NLPs; in Table 6 we report the results related to mixed complementarity problems and small VIPs.

We checked that for any single test problem, the solutions computed by all codes agree within at least ten figures.

With respect to the notation used in Tables 4, 5 and 6, we denote with $\text{size}(H_k)$ the order of matrix $H_k$ and include in brackets the order of the matrix $\hat{B}$ (i.e. $\text{size}(\hat{B})$), that we have to factorize by MINOS procedure at any step.

Furthermore, we denote with $\text{nelem}(H_k)$ the number of nonzero elements of $H_k$ and include in brackets the number of nonzero elements of the submatrix.
\( \hat{B} \) (i.e. \( \text{nelem}(\hat{B}) \)).

Moreover, \( \text{ext.} \) denotes the number of outer iterations of the non-smooth Inexact Newton method while \( \text{inn.} \) is the total number of iterations of the LSQR inner solver and \( \text{time} \) is the execution time (in seconds).

In all these Tables, we omit the execution time concerning small dimensional test problems (i.e. time smaller than one second).

In all the test problems the initial value of \( \eta_k \) in (15) is equal to 0.1 except in pinene1 \( (\eta_0 = 0.26 \times 10^{-6}) \), harkerp2 \( (\eta_0 = 0.11 \times 10^{-9}) \), ehl-kost \( (\eta_0 = 0.48 \times 10^{-6}) \)

For \( VIsemiLSQR2 \) and \( VIsemiLSQRnew2 \), we obtain a better result with respect to the total number of inner iterations and, consequently, we have a decrease in execution time because the adaptive inner stopping rule avoids unnecessary inner iterations.

Indeed in Tables 4, 5 and 6, the results obtained in \( VIsemiLSQR2 \) improve the results obtained in \( VIsemiLSQR \). Similarly, the results obtained in \( VIsemiLSQRnew2 \) improve the results obtained in \( VIsemiLSQRnew \).

This behaviour is observed for both small and large test problems.

In particular, for small scale problems we have a decrease in the number of inner iterations, and for large scale problems we also have a decrease in execution time.

The only exception is the test problem pinene1 where we do not observe better performance with respect to \( VIsemiLSQR \) and \( VIsemiLSQRnew \) with \( \eta_0 \geq 10^{-11} \).

In addition, the new right preconditioner appears very promising for the solution of large dimensional problems, as we can observe for the better results of \( VIsemiLSQRnew \) and \( VIsemiLSQRnew2 \) with respect to \( VIsemiLSQR \) and \( VIsemiLSQR2 \), above all for large dimensional problems.

We remark also that the \( VIsemiLSQRnew2 \) enables us to solve the test problem Obstacle600 in 15199 seconds \( (\text{ext} = 62, \text{inn} = 62) \).

**Acknowledgements.** The authors are very grateful to anonymous referees who stimulated us to improve the paper with their comments.
Table 2. Nonlinear programming problems

<table>
<thead>
<tr>
<th>NLP Problem</th>
<th>Ref.</th>
<th>n</th>
<th>neq</th>
<th>m</th>
<th>nl</th>
<th>nu</th>
</tr>
</thead>
<tbody>
<tr>
<td>branin</td>
<td>*</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>kowalik</td>
<td>*</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>osborne1</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>optctrl</td>
<td>1</td>
<td>28</td>
<td>19</td>
<td>1</td>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>harkerp2</td>
<td>1</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>optcodeg3</td>
<td>1</td>
<td>295</td>
<td>197</td>
<td>0</td>
<td>197</td>
<td>99</td>
</tr>
<tr>
<td>aug2dc</td>
<td>1</td>
<td>220</td>
<td>96</td>
<td>0</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>minsurf</td>
<td>5</td>
<td>225</td>
<td>0</td>
<td>0</td>
<td>225</td>
<td>0</td>
</tr>
<tr>
<td>minsurf1</td>
<td>5</td>
<td>2500</td>
<td>0</td>
<td>2500</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>minsurf2</td>
<td>5</td>
<td>3750</td>
<td>0</td>
<td>3750</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>minsurf3</td>
<td>5</td>
<td>5000</td>
<td>0</td>
<td>5000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>minsurf4</td>
<td>5</td>
<td>20000</td>
<td>0</td>
<td>20000</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>mitt105 ($\alpha = 0.01, N = 5$)</td>
<td>[14, Ex. 5.5]</td>
<td>65</td>
<td>45</td>
<td>0</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>mitt305 ($\alpha = 0.01, N = 5$)</td>
<td>[14, Ex. 4]</td>
<td>70</td>
<td>45</td>
<td>0</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>mitt549 ($\alpha = 0.01, N = 49$)</td>
<td></td>
<td>4998</td>
<td>2597</td>
<td>0</td>
<td>2401</td>
<td>4802</td>
</tr>
<tr>
<td>mitt405 ($\alpha = 0.01, N = 5$)</td>
<td>[14, Ex. 3]</td>
<td>50</td>
<td>25</td>
<td>0</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>mitt440 ($\alpha = 0.01, N = 40$)</td>
<td></td>
<td>3200</td>
<td>1600</td>
<td>0</td>
<td>1600</td>
<td>3200</td>
</tr>
<tr>
<td>mitt480 ($\alpha = 0.01, N = 80$)</td>
<td></td>
<td>12800</td>
<td>6400</td>
<td>0</td>
<td>6400</td>
<td>12800</td>
</tr>
<tr>
<td>mitt490 ($\alpha = 0.01, N = 90$)</td>
<td></td>
<td>16200</td>
<td>8100</td>
<td>0</td>
<td>8100</td>
<td>16200</td>
</tr>
<tr>
<td>mitt495 ($\alpha = 0.01, N = 95$)</td>
<td></td>
<td>18050</td>
<td>9025</td>
<td>0</td>
<td>9025</td>
<td>18050</td>
</tr>
<tr>
<td>pinene1</td>
<td>5</td>
<td>2500</td>
<td>1995</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>bearing1</td>
<td>5</td>
<td>2500</td>
<td>0</td>
<td>0</td>
<td>2500</td>
<td>0</td>
</tr>
<tr>
<td>bearing2</td>
<td>5</td>
<td>3750</td>
<td>0</td>
<td>0</td>
<td>3750</td>
<td>0</td>
</tr>
<tr>
<td>bearing3</td>
<td>5</td>
<td>5000</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>0</td>
</tr>
<tr>
<td>torsion1</td>
<td>5</td>
<td>2500</td>
<td>0</td>
<td>0</td>
<td>2500</td>
<td>2500</td>
</tr>
<tr>
<td>torsion2</td>
<td>5</td>
<td>3750</td>
<td>0</td>
<td>0</td>
<td>3750</td>
<td>3750</td>
</tr>
<tr>
<td>torsion3</td>
<td>5</td>
<td>5000</td>
<td>0</td>
<td>0</td>
<td>5000</td>
<td>5000</td>
</tr>
<tr>
<td>torsion4</td>
<td>5</td>
<td>20000</td>
<td>0</td>
<td>20000</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

*http://scicomp.ewha.ac.kr/netlib/ampl/models/nlmodels/?N=D
Table 3. Mixed complementarity problems and small variational inequality problems

<table>
<thead>
<tr>
<th>MCP Problem</th>
<th>Ref.</th>
<th>n</th>
<th>neq</th>
<th>m</th>
<th>nl</th>
<th>nu</th>
</tr>
</thead>
<tbody>
<tr>
<td>kojima-Shindo box</td>
<td>22</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Har Nach 5 box</td>
<td>22</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>PM Nasch 5 box</td>
<td>22</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Har Nasch10 box</td>
<td>22</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>bertsek</td>
<td></td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>choi</td>
<td>4</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ehl-kost</td>
<td>4</td>
<td>101</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>hydroc06</td>
<td>4</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>josephy</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>bai-haung</td>
<td>4</td>
<td>4900</td>
<td>0</td>
<td>0</td>
<td>4900</td>
<td>0</td>
</tr>
<tr>
<td>bratu</td>
<td>4</td>
<td>5929</td>
<td>0</td>
<td>0</td>
<td>5625</td>
<td>5625</td>
</tr>
<tr>
<td>Obstacle6 (N = 6)</td>
<td>22</td>
<td>36</td>
<td>0</td>
<td>0</td>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>Obstacle25 (N = 25)</td>
<td>22</td>
<td>625</td>
<td>0</td>
<td>0</td>
<td>625</td>
<td>625</td>
</tr>
<tr>
<td>Obstacle50 (N = 50)</td>
<td>22</td>
<td>2500</td>
<td>0</td>
<td>0</td>
<td>2500</td>
<td>2500</td>
</tr>
<tr>
<td>Obstacle100 (N = 100)</td>
<td>22</td>
<td>10000</td>
<td>0</td>
<td>0</td>
<td>10000</td>
<td>10000</td>
</tr>
<tr>
<td>Obstacle150 (N = 150)</td>
<td>22</td>
<td>22500</td>
<td>0</td>
<td>0</td>
<td>22500</td>
<td>22500</td>
</tr>
<tr>
<td>Obstacle200 (N = 200)</td>
<td>22</td>
<td>40000</td>
<td>0</td>
<td>0</td>
<td>40000</td>
<td>40000</td>
</tr>
<tr>
<td>Obstacle300 (N = 300)</td>
<td>22</td>
<td>90000</td>
<td>0</td>
<td>0</td>
<td>90000</td>
<td>90000</td>
</tr>
<tr>
<td>Obstacle400 (N = 400)</td>
<td>22</td>
<td>160000</td>
<td>0</td>
<td>0</td>
<td>160000</td>
<td>160000</td>
</tr>
<tr>
<td>Small VIPs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kojima-Shindo</td>
<td>22</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Braess</td>
<td>22</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Net User OPT</td>
<td>22</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Har Nach 5</td>
<td>22</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>PM Nasch 5</td>
<td>22</td>
<td>5</td>
<td>1</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Har Nasch10</td>
<td>22</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>PM Nasch10</td>
<td>22</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
References


Table 4. Results of VIsemiLSQR, VIsemiLSQR2, VIsemiLSQRnew and VIsemiLSQRnew2

<table>
<thead>
<tr>
<th>NLPs</th>
<th>size($H_k$)</th>
<th>size($\tilde{B}$)</th>
<th>nelem($H_k$)</th>
<th>nelem($\tilde{B}$)</th>
<th>ext.</th>
<th>inn.</th>
<th>time</th>
<th>ext.</th>
<th>inn.</th>
<th>time</th>
<th>ext.</th>
<th>inn.</th>
<th>time</th>
<th>ext.</th>
<th>inn.</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>branin</td>
<td>6</td>
<td>(2)</td>
<td>16</td>
<td>(4)</td>
<td>8</td>
<td>9</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>-</td>
<td>8</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>kowalik</td>
<td>12</td>
<td>(4)</td>
<td>5</td>
<td>(16)</td>
<td>37</td>
<td>51</td>
<td>-</td>
<td>37</td>
<td>37</td>
<td>-</td>
<td>22</td>
<td>22</td>
<td>-</td>
<td>22</td>
<td>22</td>
<td>-</td>
</tr>
<tr>
<td>osborne1</td>
<td>15</td>
<td>(5)</td>
<td>55</td>
<td>(25)</td>
<td>121</td>
<td>218</td>
<td>-</td>
<td>121</td>
<td>121</td>
<td>-</td>
<td>16</td>
<td>16</td>
<td>-</td>
<td>16</td>
<td>16</td>
<td>-</td>
</tr>
<tr>
<td>karkerp2</td>
<td>200</td>
<td>(100)</td>
<td>10300</td>
<td>(10000)</td>
<td>105</td>
<td>255</td>
<td>2.21</td>
<td>105</td>
<td>203</td>
<td>2.17</td>
<td>104</td>
<td>563</td>
<td>2.38</td>
<td>104</td>
<td>290</td>
<td>2.23</td>
</tr>
<tr>
<td>optdeg3</td>
<td>788</td>
<td>(492)</td>
<td>2549</td>
<td>(1763)</td>
<td>72</td>
<td>202</td>
<td>-</td>
<td>114</td>
<td>125</td>
<td>1.21</td>
<td>73</td>
<td>210</td>
<td>-</td>
<td>73</td>
<td>73</td>
<td>-</td>
</tr>
<tr>
<td>optctrl</td>
<td>78</td>
<td>(48)</td>
<td>259</td>
<td>(171)</td>
<td>34</td>
<td>78</td>
<td>-</td>
<td>32</td>
<td>32</td>
<td>-</td>
<td>26</td>
<td>57</td>
<td>-</td>
<td>26</td>
<td>26</td>
<td>-</td>
</tr>
<tr>
<td>aug2dc</td>
<td>334</td>
<td>(316)</td>
<td>1042</td>
<td>(988)</td>
<td>7</td>
<td>13</td>
<td>-</td>
<td>7</td>
<td>7</td>
<td>-</td>
<td>7</td>
<td>10</td>
<td>-</td>
<td>7</td>
<td>7</td>
<td>-</td>
</tr>
<tr>
<td>mnsurf</td>
<td>450</td>
<td>(225)</td>
<td>2132</td>
<td>(1457)</td>
<td>5</td>
<td>13</td>
<td>-</td>
<td>5</td>
<td>5</td>
<td>-</td>
<td>5</td>
<td>11</td>
<td>-</td>
<td>5</td>
<td>5</td>
<td>-</td>
</tr>
<tr>
<td>mnsurf1</td>
<td>5000</td>
<td>(2500)</td>
<td>24602</td>
<td>(17102)</td>
<td>6</td>
<td>26</td>
<td>1.56</td>
<td>6</td>
<td>6</td>
<td>1.40</td>
<td>6</td>
<td>16</td>
<td>1.42</td>
<td>6</td>
<td>6</td>
<td>1.36</td>
</tr>
<tr>
<td>mnsurf2</td>
<td>7500</td>
<td>(3750)</td>
<td>37002</td>
<td>(25752)</td>
<td>6</td>
<td>27</td>
<td>2.49</td>
<td>6</td>
<td>6</td>
<td>2.39</td>
<td>6</td>
<td>16</td>
<td>2.28</td>
<td>6</td>
<td>6</td>
<td>2.19</td>
</tr>
<tr>
<td>mnsurf3</td>
<td>10000</td>
<td>(5000)</td>
<td>49402</td>
<td>(34402)</td>
<td>6</td>
<td>26</td>
<td>3.43</td>
<td>7</td>
<td>7</td>
<td>3.84</td>
<td>6</td>
<td>16</td>
<td>3.17</td>
<td>6</td>
<td>6</td>
<td>3.05</td>
</tr>
<tr>
<td>mnsurf4</td>
<td>40000</td>
<td>(20000)</td>
<td>198802</td>
<td>(138802)</td>
<td>7</td>
<td>52</td>
<td>24.70</td>
<td>7</td>
<td>7</td>
<td>24.63</td>
<td>7</td>
<td>27</td>
<td>20.09</td>
<td>6</td>
<td>6</td>
<td>18.21</td>
</tr>
</tbody>
</table>

$^\dagger$The size of $H_k$ is equal to $n + p + m + n l + n u$ (the size of submatrix $\tilde{B}$ is $n + p + m$).

$^\ddagger$Nonzero elements of $H_k$ (nonzero elements of the submatrix $\tilde{B}$).

$^-1$Negligible execution time.
Table 5. Results of VIsemiLSQR, VIsemiLSQR2, VIsemiLSQRnew and VIsemiLSQRnew2

<table>
<thead>
<tr>
<th>NLPs</th>
<th>size($H_k$)</th>
<th>size($\hat{B}$)</th>
<th>nelem($H_k$)</th>
<th>nelem($\hat{B}$)</th>
<th>ext.</th>
<th>inn.</th>
<th>time</th>
<th>ext.</th>
<th>inn.</th>
<th>time</th>
<th>ext.</th>
<th>inn.</th>
<th>time</th>
<th>ext.</th>
<th>inn.</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>mitt349</td>
<td>14798</td>
<td>(7592)</td>
<td>56007</td>
<td>(34398)</td>
<td>2</td>
<td>30</td>
<td>2.96</td>
<td>2</td>
<td>23</td>
<td>2.82</td>
<td>2</td>
<td>30</td>
<td>2.73</td>
<td>2</td>
<td>23</td>
<td>2.70</td>
</tr>
<tr>
<td>mitt405</td>
<td>150</td>
<td>(75)</td>
<td>535</td>
<td>(310)</td>
<td>2</td>
<td>5</td>
<td>1.30</td>
<td>2</td>
<td>4</td>
<td>1.02</td>
<td>2</td>
<td>4</td>
<td>0.84</td>
<td>2</td>
<td>4</td>
<td>0.82</td>
</tr>
<tr>
<td>mitt440</td>
<td>9600</td>
<td>(4800)</td>
<td>36480</td>
<td>(22080)</td>
<td>2</td>
<td>18</td>
<td>1.30</td>
<td>2</td>
<td>14</td>
<td>1.27</td>
<td>2</td>
<td>16</td>
<td>1.14</td>
<td>2</td>
<td>14</td>
<td>1.11</td>
</tr>
<tr>
<td>mitt480</td>
<td>38400</td>
<td>(19200)</td>
<td>146560</td>
<td>(88960)</td>
<td>2</td>
<td>41</td>
<td>13.57</td>
<td>2</td>
<td>40</td>
<td>11.24</td>
<td>2</td>
<td>42</td>
<td>11.43</td>
<td>2</td>
<td>39</td>
<td>11.24</td>
</tr>
<tr>
<td>mitt490</td>
<td>48600</td>
<td>(24300)</td>
<td>185580</td>
<td>(112680)</td>
<td>2</td>
<td>49</td>
<td>27.55</td>
<td>2</td>
<td>46</td>
<td>20.17</td>
<td>2</td>
<td>50</td>
<td>19.16</td>
<td>2</td>
<td>49</td>
<td>16.26</td>
</tr>
<tr>
<td>mitt:495</td>
<td>54150</td>
<td>(27075)</td>
<td>206815</td>
<td>(125590)</td>
<td>2</td>
<td>58</td>
<td>20.58</td>
<td>2</td>
<td>48</td>
<td>20.37</td>
<td>2</td>
<td>60</td>
<td>19.10</td>
<td>2</td>
<td>48</td>
<td>17.08</td>
</tr>
<tr>
<td>pinene1</td>
<td>4000</td>
<td>(3995)</td>
<td>38337</td>
<td>(38322)</td>
<td>73</td>
<td>8439</td>
<td>94.80</td>
<td>*</td>
<td>*</td>
<td>66</td>
<td>7335</td>
<td>82.76</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>bearing1</td>
<td>5000</td>
<td>(2500)</td>
<td>19800</td>
<td>(12300)</td>
<td>11</td>
<td>36</td>
<td>2.10</td>
<td>11</td>
<td>11</td>
<td>1.95</td>
<td>11</td>
<td>30</td>
<td>2.02</td>
<td>11</td>
<td>11</td>
<td>1.92</td>
</tr>
<tr>
<td>bearing2</td>
<td>7500</td>
<td>(3750)</td>
<td>29750</td>
<td>(18500)</td>
<td>12</td>
<td>39</td>
<td>3.75</td>
<td>12</td>
<td>12</td>
<td>3.50</td>
<td>12</td>
<td>34</td>
<td>3.71</td>
<td>12</td>
<td>32</td>
<td>3.53</td>
</tr>
<tr>
<td>bearing3</td>
<td>10000</td>
<td>(5000)</td>
<td>39700</td>
<td>(24700)</td>
<td>13</td>
<td>45</td>
<td>5.82</td>
<td>13</td>
<td>13</td>
<td>5.69</td>
<td>13</td>
<td>39</td>
<td>5.79</td>
<td>13</td>
<td>13</td>
<td>5.68</td>
</tr>
<tr>
<td>torsion1</td>
<td>7500</td>
<td>(2500)</td>
<td>27300</td>
<td>(12300)</td>
<td>21</td>
<td>57</td>
<td>4.24</td>
<td>21</td>
<td>21</td>
<td>3.99</td>
<td>18</td>
<td>47</td>
<td>2.88</td>
<td>18</td>
<td>18</td>
<td>2.84</td>
</tr>
<tr>
<td>torsion2</td>
<td>11250</td>
<td>(3750)</td>
<td>41000</td>
<td>(18500)</td>
<td>24</td>
<td>68</td>
<td>7.88</td>
<td>24</td>
<td>24</td>
<td>7.40</td>
<td>21</td>
<td>58</td>
<td>5.43</td>
<td>21</td>
<td>21</td>
<td>5.13</td>
</tr>
<tr>
<td>torsion3</td>
<td>15000</td>
<td>(5000)</td>
<td>54700</td>
<td>(24700)</td>
<td>30</td>
<td>91</td>
<td>13.80</td>
<td>30</td>
<td>30</td>
<td>12.91</td>
<td>24</td>
<td>66</td>
<td>8.60</td>
<td>24</td>
<td>24</td>
<td>8.13</td>
</tr>
<tr>
<td>torsion4</td>
<td>60000</td>
<td>(20000)</td>
<td>219400</td>
<td>(99400)</td>
<td>49</td>
<td>202</td>
<td>137.41</td>
<td>49</td>
<td>49</td>
<td>125.24</td>
<td>39</td>
<td>137</td>
<td>86.92</td>
<td>39</td>
<td>39</td>
<td>80.01</td>
</tr>
</tbody>
</table>

1The size of $H_k$ is equal to $n + p + m + nl + nu$ (the size of submatrix $\hat{B}$ is $n + p + m$).

2Nonzero elements of $H_k$ (nonzero elements of the submatrix $\hat{B}$).

3Negligible execution time.
<table>
<thead>
<tr>
<th>MCF's</th>
<th>size($\tilde{H}_d$)</th>
<th>size($\tilde{B}$)</th>
<th>nelem($\tilde{H}_d$)</th>
<th>nelem($\tilde{B}$)</th>
<th>ext. inn. time</th>
<th>ext. inn. time</th>
<th>ext. inn. time</th>
<th>ext. inn. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>bertsekas-Shindo box</td>
<td>25</td>
<td>15</td>
<td>90</td>
<td>60</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>Har Nach. 5 box</td>
<td>10</td>
<td>5</td>
<td>40</td>
<td>25</td>
<td>7</td>
<td>14</td>
<td>7</td>
<td>14</td>
</tr>
<tr>
<td>Har Nach. 10 box</td>
<td>20</td>
<td>10</td>
<td>80</td>
<td>40</td>
<td>18</td>
<td>35</td>
<td>18</td>
<td>35</td>
</tr>
<tr>
<td>bai-haung</td>
<td>9800</td>
<td>9800</td>
<td>38920</td>
<td>38920</td>
<td>6</td>
<td>13</td>
<td>6</td>
<td>13</td>
</tr>
<tr>
<td>Bratu</td>
<td>17179</td>
<td>17179</td>
<td>63179</td>
<td>63179</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Net User OPT</td>
<td>12 (7)</td>
<td>12 (7)</td>
<td>36 (21)</td>
<td>36 (21)</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Obstacle25</td>
<td>13 (5)</td>
<td>13 (5)</td>
<td>39 (23)</td>
<td>39 (23)</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Obstacle30</td>
<td>10 (5)</td>
<td>10 (5)</td>
<td>30 (18)</td>
<td>30 (18)</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Obstacle150</td>
<td>67500</td>
<td>67500</td>
<td>439200</td>
<td>439200</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Obstacle400</td>
<td>480000</td>
<td>480000</td>
<td>1758400</td>
<td>1758400</td>
<td>18</td>
<td>18</td>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>

The size of $\tilde{B}$ is equal to $n + p + m + n (m + n)$ (the size of submatrix $\tilde{B}$).