Operator Splitting Methods for Monotone Affine Variational Inequalities, with a Parallel Application to Optimal Control*

Jonathan Eckstein†
Michael C. Ferris‡

July 30, 1996

Abstract

This paper applies splitting techniques developed for set-valued maximal monotone operators to monotone affine variational inequalities, including as a special case the classical linear complementarity problem. We give a unified presentation of several splitting algorithms for monotone operators, and then apply these results to obtain two classes of algorithms for affine variational inequalities. The second class resembles classical matrix splitting, but has a novel "under-relaxation" step, and converges under more general conditions. In particular, the convergence proofs do not require the affine operator to be symmetric. We specialize our matrix-splitting-like method to discrete-time optimal control problems formulated as extended linear-quadratic programs in the manner advocated by Rockafellar and Wets. The result is a highly parallel algorithm, which we implement and test on the Connection Machine CM–5 computer family.

The *affine variational inequality problem* is to find a vector x lying in a closed convex set $\mathcal{B}$ such that

$$\langle Mx + q, w - x \rangle \geq 0 \quad \forall\, w \in \mathcal{B}\ , \tag{1}$$

where M is a given $n \times n$ matrix and q is a vector from $\Re^n$. We denote this problem $\text{avi}(M, q, \mathcal{B})$. A common and important special case occurs when $\mathcal{B}$ is a box, that is,

$$\mathcal{B} = \prod_{i=1}^{n} \{x_i \in \Re \mid \ell_i \leq x_i \leq u_i\}\ , \tag{2}$$

with $\ell_i \in [-\infty, \infty)$, $u_i \in (-\infty, \infty]$, and $\ell_i \leq u_i$. It is well known that when $\ell \equiv 0$ and $u \equiv \infty$, the problem reduces to the *linear complementarity problem* (LCP) [5, 6, 24] of finding some $x \in \Re^n$ satisfying

$$x \geq 0 \qquad Mx + q \geq 0 \qquad \langle x, Mx + q \rangle = 0\ . \tag{3}$$

This paper is restricted to the *monotone* case of $\text{avi}(M, q, \mathcal{B})$, where M is positive semidefinite, although not necessarily symmetric.

*This work is partially supported by the National Science Foundation under grants CCR-9157632 and CDA-9024618, the Air Force Office of Scientific Research under grant F49620-94-1-0036 and the Department of Energy grant DE-FG03-94ER61915.

†School of Business and RUTCOR, Rutgers University, New Brunswick, NJ 08903

‡Computer Sciences Department, University of Wisconsin, Madison, WI 53706

We propose to solve such problems with algorithms derived from the splitting theory of maximal monotone operators. Section 1 reviews relevant portions of this theory in a tutorial manner, although it contains some minor new results that are used later in the paper. It is distinguished from prior expositions (*e.g.* [20] and [11, Chapter 3]) in that it emphasizes the fundamental connection between monotone operators and nonexpansive mappings, inspired by [17]. We feel this approach is more intuitive than treatments based on proximal mappings, and it serves to unify the treatment of "Douglas-Rachford" and "Peaceman-Rachford" schemes [20, 12].

Section 2 applies the theory in two different ways to monotone affine variational inequality (MAVI) problems and derives two classes of MAVI algorithms. The first class is quite simple and is not entirely new, overlapping with special cases of the dual methods proposed in [14]. Despite their simplicity, these methods have remained virtually unknown; for example, they are absent from recent reference works such as [6]. We show that they take a very simple form indeed in the case of the LCP (3); see (27) below.

The second class of MAVI methods we derive appears to be entirely new, although it resembles classical matrix splitting [26, 27, 28], with the usual under-relaxation step replaced by the solution of a certain linear system. The new class of methods has a stronger convergence theory that does not depend on symmetry.

Section 3 takes an algorithm from this latter class and applies it to discrete-time optimal control problems formulated as extended linear-quadratic programs [40, 41] (see also [7, Section 3.6]). The purpose of this exercise is twofold: first, to demonstrate the methods of Section 2 are not merely theoretical, but can be successfully applied to difficult, large-scale problems; second, to provide an example of how splitting methods can be used to produce parallel algorithms. Essentially, if one can express a problem as the superposition of two structures, each by itself amenable to parallel computing, then splitting will furnish an iterative parallel algorithm that deals alternately with one structure and then the other.

We then describe a massively data-parallel implementation of our proposed splitting method for extended linear-quadratic programming on the CM–5 family of parallel computers. Computational results show that the parallel splitting implementation greatly outperforms standard serial codes on all but the smallest test problems.

Section 4 gives some concluding remarks.

1 Summary of Monotone Operator Splitting Theory

1.1 Monotone Operators

In this paper, an *operator* T on $\Re^n$ is simply some subset of $\Re^n \times \Re^n$. For every such T, we let $T(x) \doteq \{y \mid (x,y) \in T\}$, thus defining a point-to-set mapping on $\Re^n$; in fact, we make no distinction between this point-to-set mapping and its graph T. Thus, the statements $y \in T(x)$ and $(x,y) \in T$ are completely equivalent.

The *inverse* of any operator T is the operator $T^{-1} = \{(y,x) \mid (x,y) \in T\}$, which will always exist. The sum $T_1 + T_2$ of two operators T_1 and T_2 is defined by

$$T(x) = (T_1 + T_2)(x) = T_1(x) + T_2(x) \doteq \{y + z \mid y \in T_1(x), z \in T_2(x)\} \quad , \tag{4}$$

and for any $c \in \Re$, $cT \doteq \{(x, cy) \mid (x,y) \in T\}$. We define $\operatorname{dom} T \doteq \{x \mid T(x) \neq \emptyset\}$ and $\operatorname{im} T \doteq \operatorname{dom}(T^{-1}) = \{y \mid \exists\, (x,y) \in T\}$. When $T(x)$ is necessarily a singleton set $\{y\}$ for all x, that is, T is the graph of a function $\Re^n \to \Re^n$, we say that T is *single-valued*, and we may write $T(x) = y$ instead of $T(x) = \{y\}$.

An operator T is said to be *monotone* if

$$\langle x - x', y - y' \rangle \geq 0 \quad \forall (x,y),(x',y') \in T \quad . \tag{5}$$

A monotone operator T is said to be *maximal* if no strict superset of T is monotone, that is, $T \subseteq T' \subseteq \Re^n \times \Re^n$ with T' monotone implies that $T' = T$.

Finding a *zero* of a maximal monotone operator T, that is, some $x \in \Re^n$ with $0 \in T(x)$, is a fundamental problem [1, 30, 39] that generalizes not only all of lower semicontinuous convex optimization, but also monotone variational inequalities.

Some salient facts regarding monotone and maximal monotone operators on $\Re^n$ are collected below.

Proposition 1

1. *For any scalar $c > 0$, cT is (maximal) monotone if and only if T is (maximal) monotone; T and cT have the same set of zeroes.*

2. *Suppose T is monotone, then T is maximal monotone if and only if* $\mathrm{im}(I + T) = \Re^n$.

3. *Given any monotone operator T on $\Re^n$ and $z \in \Re^n$, z can be written in at most one way as $x + y$, where $y \in T(x)$. If T is maximal, then such a representation must exist.*

4. *If $f\colon \Re^n \to \Re \cup \{+\infty\}$ is a lower semi-continuous convex function, then the subgradient operator ∂f (see for example [36]) is maximal monotone.*

5. *If T_1 and T_2 are monotone operators, then $T_1 + T_2$ is also monotone. If T_1 and T_2 are both maximal and* $\mathrm{ri}(\mathrm{dom}\ T_1) \cap \mathrm{ri}(\mathrm{dom}\ T_2) \neq \emptyset$, *where "*ri*" denotes the relative interior (see for example [36]), then $T_1 + T_2$ is also maximal.*

6. *If Q is an invertible matrix, then the operator*

$$Q^\top T Q \doteq \left\{ (Q^{-1}x, Q^\top y) \mid (x,y) \in T \right\}$$

is (maximal) monotone if and only if T is (maximal) monotone.

Proof. Statement 1 is trivial. Statement 2 is shown in [1, 23]. For the first part of statement 3, suppose $z = x + y = x' + y'$, where $(x,y),(x',y') \in T$. Then $0 \leq \langle x - x', y - y' \rangle = \langle x - x', (z - x) - (z - x') \rangle = -\|x - x'\|^2$, implying $x = x'$ and $y = y'$. The rest of the statement follows from statement 2 (see also [23] for an equivalent result). The next statement is standard and can be found in [35, 36]. The result about the maximality of sums of operators is proven in [37]. Statement 6 can be established directly or may be found in [11, Proposition 3.1(iv)]. $\square$

We now consider another, more familiar class of operators. An operator N on $\Re^n$ is said to be *nonexpansive* if

$$\|w - w'\| \leq \|z - z'\| \quad \forall (z,w),(z',w') \in N \quad . \tag{6}$$

Considering the case $z = z'$, it is immediate that nonexpansive operators must be single-valued, and (6) may be simplified to

$$\|N(z) - N(z')\| \leq \|z - z'\| \quad \forall z, z' \in \Re^n \quad . \tag{7}$$

It is also immediate that nonexpansive operators are Lipschitz continuous, and a composition of nonexpansive mappings is nonexpansive.

We now develop some observations from [17, 23] that reveal a deep connection between monotone and nonexpansive operators. Let T be any operator and consider the related operator $\mathcal{N}[T]$ given by

$$\mathcal{N}[T] = \{(x+y, x-y) \mid (x,y) \in T\} \quad .$$

This transformation is invertible, via

$$\mathcal{N}^{-1}[N] = \left\{ \left(\frac{z+w}{2}, \frac{z-w}{2} \right) \mid (z,w) \in N \right\} \quad ,$$

so that $\mathcal{N}^{-1}[\mathcal{N}[T]] = T$.

Proposition 2 *Suppose $N = \mathcal{N}[T]$ (or equivalently $T = \mathcal{N}^{-1}[N]$). Then T is monotone if and only if N is nonexpansive. T is maximal monotone if and only if N is a nonexpansive mapping defined on all of $\Re^n$. Finally, $0 \in T(x)$ if and only if x is a fixed point of N, that is, $x \in N(x)$.*

Proof. Choose any $(x,y), (x',y') \in T$ and $(z,w), (z',w') \in N$ related by $x+y=z$, $x-y=w$, $x'+y'=z'$, and $x'-y'=w'$. Then

$$\begin{aligned} \|w-w'\|^2 &= \|(x-y)-(x'-y')\|^2 \\ &= \|(x+y)-(x'+y')\|^2 - 4\langle x-x', y-y'\rangle \\ &= \|z-z'\|^2 - 4\langle x-x', y-y'\rangle \quad . \end{aligned} \tag{8}$$

It follows that N is nonexpansive if and only if $\langle x-x', y-y'\rangle \geq 0$ for all $(x,y), (x',y') \in T$. This proves the first equivalence. Combining this observation with Proposition 1, statement 2, we obtain the second equivalence. For the last assertion, note that

$$0 \in T(x) \quad \Leftrightarrow \quad (x+0, x-0) = (x,x) \in N \quad .$$

□

Thus, for every monotone operator T, there is a corresponding nonexpansive mapping $N = \mathcal{N}[T]$. The zeroes of T are the fixed points of N. Conversely, for every nonexpansive mapping N, there is a corresponding monotone operator $T = \mathcal{N}^{-1}[N]$ whose zeroes are the fixed points of N.

This equivalence suggests that an iteration of the form $z^{k+1} = \mathcal{N}[T](z^k)$ might be able to locate the zeroes of T. Unfortunately, the nonexpansiveness of $\mathcal{N}[T]$ does not guarantee convergence of such a $\{z^k\}$, but only that it will remain within a bounded distance of any solution point. Furthermore, $\mathcal{N}[T](z)$ may be difficult to evaluate, since it involves finding the unique $(x,y) \in T$ with $x+y=z$, which is in many cases as hard as solving $T(x) \ni 0$. Fortunately, it is possible to apply the following proposition, where $a \underset{\delta}{\approx} b$ is a shorthand for $\|a-b\| \leq \delta$:

Proposition 3 *Let N be any nonexpansive mapping defined on all of $\Re^n$, possessing at least one fixed point. Let the sequence $\{z^k\} \subset \Re^n$ conform to the recursion*

$$z^{k+1} \underset{\delta_k}{\approx} \lambda_k N(z^k) + (1-\lambda_k)z^k \quad , \tag{9}$$

where $\{\lambda_k\}$ is a sequence of scalars with $0 < \lambda_k \le 1$. If $0 < \inf\{\lambda_k\}_{k=0}^{\infty} \le \sup\{\lambda_k\}_{k=0}^{\infty} < 1$ and $\sum_{i=0}^{\infty} \delta_k < \infty$ then $\{z^k\}$ converges to a fixed point of N. If N is a contraction, that is, there exists some $\alpha \in [0,1)$ such that

$$\|N(z) - N(z')\| \le \alpha \|z - z'\| \quad \forall z, z' \in \Re^n \quad , \tag{10}$$

then convergence is guaranteed under the weaker assumption that $\sum_{k=0}^{\infty} \lambda_k = \infty$ and $\frac{\delta_k}{\lambda_k} \to 0$.

Proof. In view of Proposition 2, the first case is implied by [12, Theorem 3] applied to the maximal monotone operator $T = \mathcal{N}^{-1}[N]$, where $\rho_k = 2\lambda_k$ and $c_k = 1$ for all k (note: this theorem generalizes many prior results, including [25, Theorem 3] and [38, Theorem 1]).

The second case follows from direct application of [33, Lemma 3, page 45]. □

Note that

$$\tfrac{1}{2}I + \tfrac{1}{2}\mathcal{N}[T] = \{(x+y, x) \mid (x,y) \in T\} = (I+T)^{-1} \quad .$$

Thus, if we leave λ_k fixed at $1/2$, we are executing the iteration

$$z^{k+1} \underset{\delta_k}{\approx} (I+T)^{-1}(z^k) \quad ,$$

which is a form of the well-known *proximal point algorithm* [38]. The general form of the proximal point algorithm differs only in allowing iterations of the form

$$z^{k+1} \underset{\delta_k}{\approx} (I + c_k T)^{-1}(z^k) \quad ,$$

where $\{c_k\}$ is a sequence of positive scalars bounded away from zero.

If $\sup\{\lambda_k\}_{k=0}^{\infty} = 1$, Proposition 3 requires N to be a contraction. The following lemma gives sufficient conditions on T for this to occur for $N = \mathcal{N}[T]$.

Lemma 4 *Suppose the operator T is strongly monotone with modulus σ, and Lipschitz with modulus ω, that is, there exist some $\sigma, \omega > 0$ such that for all $(x,y), (x',y') \in T$,*

$$\begin{aligned} \langle x - x', y - y' \rangle \quad &\ge \quad \sigma \|x - x'\|^2 && (11) \\ \|y - y'\| \quad &\le \quad \omega \|x - x'\| \quad . && (12) \end{aligned}$$

Then $N = \mathcal{N}[T]$ is a contraction.

Proof. Applying the Cauchy-Schwarz inequality, (11) implies

$$\|y - y'\| \ge \sigma \|x - x'\| \qquad \forall\, (x,y), (x',y') \in T \qquad .$$

It follows that $\sigma \le \omega$. Let

$$\alpha = \sqrt{\frac{1 - 2\sigma + \omega^2}{1 + 2\sigma + \omega^2}} \quad . \tag{13}$$

Since $\sigma > 0$, it follows that the denominator in (13) is positive. Furthermore, since $0 < \sigma \le \omega$, we have $1 - 2\sigma + \omega^2 \ge (1-\omega)^2 \ge 0$, so the numerator is nonnegative and $\alpha \in [0,1)$.

Substituting the definition of α on both sides now establishes the validity of the equation

$$2\sigma(\alpha^2 + 1) = (1 - \alpha^2)(1 + \omega^2) \quad . \tag{14}$$

Take any $(x,y),(x',y') \in T$. Multiplying both sides of (14) by $\|x-x'\|^2$, we have

$$2\sigma(\alpha^2+1)\|x-x'\|^2 = (1-\alpha^2)(\|x-x'\|^2 + \omega^2\|x-x'\|^2) \quad . \tag{15}$$

Since T is Lipschitz, (15) implies

$$2\sigma(\alpha^2+1)\|x-x'\|^2 \geq (1-\alpha^2)(\|x-x'\|^2 + \|y-y'\|^2) \quad . \tag{16}$$

By the strong monotonicity of T, (16) implies

$$2(\alpha^2+1)\langle x-x', y-y'\rangle \geq (1-\alpha^2)(\|x-x'\|^2 + \|y-y'\|^2) \quad ,$$

or equivalently

$$\|x-x'\| - 2\langle x-x', y-y'\rangle + \|y-y'\|^2 \leq \alpha^2\left[\|x-x'\| + 2\langle x-x', y-y'\rangle + \|y-y'\|^2\right] \quad ,$$

which is in turn equivalent to

$$\|(x-y)-(x'-y')\|^2 \leq \alpha^2\|(x+y)-(x'+y')\|^2 \quad . \tag{17}$$

Now, statement 3 of Proposition 1 states that every z in the domain of N may be expressed as $x+y$ for some $(x,y)\in T$, in which case $N(z) = x-y$. Therefore, taking square roots, (17) implies $\|N(z)-N(z')\| \leq \alpha\|z-z'\|$ for every $z, z' \in \operatorname{dom} N$. □

Conditions (11)-(12) are the same as those employed in the rate of convergence results in works such as [20] and [21].

1.2 Rachford-Class Splitting Theory

In many cases, the operator $\mathcal{N}[T]$, or essentially equivalently $(I+T)^{-1}$, may be too difficult to evaluate to make direct application of Proposition 3 practical. Suppose that this is the case, but that T has the special structure $T = T_1 + T_2$ given in (4), where T_1 and T_2 are both maximal monotone but have simple enough structure to permit practical evaluation of both $N_1 \doteq \mathcal{N}[T_1]$ and $N_2 \doteq \mathcal{N}[T_2]$.

Now, N_1 and N_2 defined in this manner are both nonexpansive maps, so their functional composition $\overline{N} \doteq N_1 \circ N_2 = \mathcal{N}[T_1] \circ \mathcal{N}[T_2]$ is also nonexpansive. Furthermore, the following property holds:

Lemma 5 *Let T_1, T_2 be maximal monotone operators on $\Re^n$. Then $z \in \Re^n$ is a fixed point of $\overline{N} = \mathcal{N}[T_1]\circ\mathcal{N}[T_2]$ if and only if it is of the form $x+y$, where $y \in T_2(x)$ and $-y \in T_1(x)$. This property in turn implies that $x = \frac{1}{2}z + \frac{1}{2}\mathcal{N}[T_2](z) = (I+T_2)^{-1}(z)$ is a zero of T_1+T_2.*

Proof. First suppose $z = x+y$, $y \in T_2(x)$, and $-y \in T_1(x)$. Then $\overline{N}(z) = \mathcal{N}[T_1](\mathcal{N}[T_2](z)) = \mathcal{N}[T_1](x-y) = x-(-y) = x+y = z$. Conversely, suppose z is a fixed point of $\overline{N}$. Then there must exist $(x,y) \in T_2$ with $x+y = z$. Then $\mathcal{N}[T_2](z) = x-y$. Similarly, there must exist $(x',y') \in T_1$ such that $x'+y' = x-y$. Then $\mathcal{N}[T_1](x-y) = x'-y'$. By hypothesis, $x'-y' = z = x+y$, so we have the system of equations

$$\begin{aligned} x'+y' &= x-y \\ x'-y' &= x+y \quad . \end{aligned}$$

Adding these two equations yields $x' = x$, and subtracting them yields $y' = -y$. It follows that z is of the specified form and that $T_1(x)+T_2(x) \ni -y+y = 0$, implying that x is a zero of T_1+T_2. □

Thus, finding a fixed point of $\overline{N}$ is tantamount to finding a zero of $T = T_1 + T_2$. As $\overline{N}$ is nonexpansive, we can directly apply Proposition 3. For simplicity of notation, suppose that $\delta_k \equiv 0$. The iteration suggested by Proposition 3,

$$\begin{aligned} z^{k+1} &= \lambda_k \overline{N}(z^k) + (1-\lambda_k)(z^k) && (18) \\ &= \lambda_k N_1(N_2(z^k)) + (1-\lambda_k)(z^k) \quad , && (19) \end{aligned}$$

can be efficiently carried out as follows. Let $(x^k, b^k) \in T_2$ be such that $z^k = x^k + b^k$ for all k and iterate using the ensuing two steps:

(i) Find the unique $(y^{k+1}, a^{k+1}) \in T_1$ such that

$$y^{k+1} + a^{k+1} = x^k - b^k \quad . \tag{20}$$

(ii) Find the unique $(x^{k+1}, b^{k+1}) \in T_2$ such that

$$\begin{aligned} x^{k+1} + b^{k+1} &= \lambda_k(y^{k+1} - a^{k+1}) + (1-\lambda_k)(x^k + b^k) \\ &= \lambda_k\left(y^{k+1} - (x^k - b^k - y^{k+1})\right) + (1-\lambda_k)(x^k + b^k) \\ &= 2\lambda_k y^{k+1} + (1-2\lambda_k)x^k + b^k \quad . \end{aligned}$$

The case $\lambda_k \equiv 1$ is known as "Peaceman-Rachford" splitting, while the case $\lambda_k \equiv 1/2$ is traditionally known as "Douglas-Rachford" splitting [20]. The more general case $0 < \inf\{\lambda_k\}_{k=0}^{\infty} \leq \sup\{\lambda_k\}_{k=0}^{\infty} < 1$ is addressed in [12].

If one allows Peaceman-Rachford steps, that is, one permits λ_k to equal or approach 1, then $\overline{N}$ must be a contraction for Proposition 3 to guarantee convergence. Such a property can be guaranteed if either $N_1 = \mathcal{N}[T_1]$ or $N_2 = \mathcal{N}[T_2]$ is a contraction, and in particular, by Lemma 4, if either T_1 or T_2 is both strongly monotone and Lipschitz.

We also note that Proposition 3 will permit approximate evaluation of $\overline{N}$ in (18), which means that either N_1 and/or N_2 may be evaluated approximately. We thus arrive at the following proposition:

Proposition 6 *Let T_1, T_2 be maximal monotone operators on $\Re^n$ such that $0 \in T_1(x) + T_2(x)$ has some solution. Let $\{\alpha_k\}, \{\beta_k\}, \{\lambda_k\}$ be sequences of nonnegative scalars with $\epsilon_1 \leq \lambda_k \leq 1 - \epsilon_2$, where $\epsilon_1 > 0$ and $\epsilon_2 \geq 0$. Suppose $\{(x^k, b^k)\} \subset T_2$ and $\{(y^k, a^k)\} \subset T_1$ conform to the recursions*

$$\begin{aligned} y^{k+1} + a^{k+1} &\underset{\alpha_k}{\approx} x^k - b^k && (21) \\ x^{k+1} + b^{k+1} &\underset{\beta_k}{\approx} 2\lambda_k y^{k+1} + (1-2\lambda_k)x^k + b^k && (22) \end{aligned}$$

for all $k \geq 0$. Then, if either of the following two conditions hold, $\{x^k\}$ converges to some solution x^* of $0 \in T_1(x) + T_2(x)$:

1. *$\epsilon_2 > 0$, $\sum_{k=0}^{\infty} \alpha_k < \infty$, and $\sum_{k=0}^{\infty} \beta_k < \infty$.*

2. *One of T_1 or T_2 is both strongly monotone and Lipschitz, $\beta_k/\lambda_k \to 0$, and $\alpha_k \to 0$.*

Proof. Let $z^k = x^k + b^k$ and $w^k = y^k + a^k$ for all $k \geq 0$. Then, defining N_1, N_2 as above, we have

$$\begin{aligned} w^k &\underset{\alpha_k}{\approx} N_2(z^k) \\ z^{k+1} &\underset{\beta_k}{\approx} \lambda_k N_1(w^k) + (1-\lambda_k)\, z^k \quad , \end{aligned}$$

which implies by the nonexpansiveness of N_2 that

$$z^{k+1} \underset{\beta_k+\lambda_k\alpha_k}{\approx} \lambda_k N_1(N_2(z^k)) + (1-\lambda_k)z^k \quad .$$

First, consider case 1. Since $\{\alpha^k\}$ and $\{\beta_k\}$ are summable, and $\{\lambda_k\}$ is bounded, $\{\beta_k + \lambda_k\alpha_k\}$ is also summable. Therefore, Proposition 3 implies that $\{z^k\}$ converges to a fixed point of the nonexpansive map $N_1 \circ N_2 = \overline{N}$, that is, some $x^* + b^*$ such that $b^* \in T_2(x^*)$, $-b^* \in T_1(x^*)$, and thus $0 \in T_1(x^*) + T_2(x^*)$. Now, $x^k = (1/2)(N_2(z^k) + z_k)$ for all k; since N_2 is nonexpansive, it is Lipschitz continuous, and hence $\lim_{k\to\infty} x^k = \lim_{k\to\infty} \frac{1}{2}\left(N_2(z^k) + z^k\right) = \frac{1}{2}N_2(z^*) + \frac{1}{2}z^* = x^*$.

Now consider case 2. As outlined above, $\overline{N}$ is now guaranteed to be a contraction. $\beta_k/\lambda_k \to 0$ and $\alpha_k \to 0$ collectively imply that $(\beta_k + \lambda_k\alpha_k)/\lambda_k = \beta_k/\lambda_k + \alpha_k \to 0$. Therefore, we can apply the contraction case of Proposition 3, and proceed as in case 1. □

Note that the theory we have just presented covers only Peaceman- and Douglas-Rachford splitting, and does not subsume forward-backward [4, 14, 31, 49], or "double-backward" [19, 31, 22] splitting schemes. These methods have a different and generally less attractive convergence theory.

2 Relating Monotone Affine Variational Inequalities to Monotone Operators

We now relate the monotone operator theory we have just presented to monotone affine variational inequality problems. Recall the problem $\mathrm{avi}(M, q, \mathcal{B})$ given in (1), with M positive semidefinite but possibly asymmetric, and let $\mathrm{sol}(M, q, \mathcal{B})$ denote its solution or, if there is more than one, the set of all solutions. The *feasible region* of $\mathrm{avi}(M, q, \mathcal{B})$ is defined as

$$\mathrm{feas}(M, q, \mathcal{B}) \doteq \{x \mid Mx + q \in (\mathrm{rec}\,\mathcal{B})^*, x \in \mathcal{B}\},$$

where $\mathrm{rec}\,C$ denotes the recession cone of a set C defined by

$$\mathrm{rec}\,C \doteq \{d \in \Re^n \mid x + cd \in C \; \forall\, x \in C, c \geq 0\} \;,$$

and "*" denotes the dual cone operation defined by

$$K^* \doteq \{y \mid \langle y, v\rangle \geq 0, \forall\, v \in K\}.$$

By way of illustration, in the special case of the LCP (3) the feasible set is $\{x \mid Mx + q \geq 0, x \geq 0\}$, while the solution set consists of all elements of the feasible set that also satisfy the complementarity condition $\langle x, Mx + q\rangle = 0$.

2.1 Simple Splitting Schemes

We now introduce two operators that constitute splitting of $\mathrm{avi}(M, q, \mathcal{B})$. Let W denote the operator

$$W(x) \doteq \{Mx + q\} \quad \forall x \in \Re^n \;.$$

It is simple to confirm that an operator of this form must be Lipschitz, is monotone if and only if M is positive semidefinite, and is strongly monotone if and only if M is positive definite. Furthermore, Proposition 1 implies W is maximal, because $I + M$ is positive definite, and hence

$$\mathrm{im}\,(I + W) = \{(I + M)x + q \mid x \in \Re^n\} = \Re^n \;.$$

Let $N_{\mathcal{B}}$ denote the point-to-set *normal cone operator* of $\mathcal{B}$ defined by

$$N_{\mathcal{B}}(x) \doteq \begin{cases} \{y \mid \langle y, w-x\rangle \le 0 \ \forall w \in \mathcal{B}\}, & x \in \mathcal{B} \\ \emptyset, & x \notin \mathcal{B} . \end{cases}$$

Since $\mathcal{B}$ is closed and convex, $N_{\mathcal{B}}$ is the subgradient mapping of the closed proper convex function $\delta_{\mathcal{B}}: \Re^n \to \Re \cup \{+\infty\}$ given by

$$\delta_{\mathcal{B}}(x) \doteq \begin{cases} 0, & x \in \mathcal{B} \\ +\infty, & x \notin \mathcal{B} . \end{cases}$$

$N_{\mathcal{B}}$ is therefore maximal monotone by Proposition 1, statement 4.

The problem $\operatorname{avi}(M, q, \mathcal{B})$ is equivalent to requiring that $-(Mx+q) \in N_{\mathcal{B}}(x)$, that is,

$$0 \in Mx + q + N_{\mathcal{B}}(x) . \tag{23}$$

By the above definitions this is equivalent to finding some zero x of the operator $W + N_{\mathcal{B}}$, namely solving $0 \in (W + N_{\mathcal{B}})(x)$. Since $\operatorname{dom} W = \Re^n$, $W + N_{\mathcal{B}}$ is maximal monotone by Proposition 1, statement 5.

We may therefore envision solving MAVI's by applying one of the splitting schemes outlined in Section 1, setting

$$\begin{aligned} &T_1(x) = W(x) = \{Mx + q\} \qquad \forall x \in \Re^n, \\ &T_2 = N_{\mathcal{B}} . \end{aligned}$$

Generalizing slightly, we introduce a positive scalar multiplier c:

$$\begin{aligned} &T_1(x) = cW(x) = \{cMx + cq\} \qquad \forall x \in \Re^n, \\ &T_2 = cN_{\mathcal{B}} = N_{\mathcal{B}} . \end{aligned} \tag{24}$$

Consider applying Proposition 6 to the identifications (24). For simplicity, we will let $\alpha_k = \beta_k = 0$ for all k. For all $v \in \Re^n$, define $(v)_{\mathcal{B}}$ to be the projection of v onto $\mathcal{B}$. A critical observation is that

$$z = x + b, \ (x, b) \in N_{\mathcal{B}} \iff z = (x)_{\mathcal{B}}, \ b = z - x . \tag{25}$$

In this case, it follows that

$$x - b = x - (z - x) = 2\,(z)_{\mathcal{B}} - z \doteq r_{\mathcal{B}}(x) .$$

The operation $r_{\mathcal{B}}$ is a reflection through the set $\mathcal{B}$.

To apply the iteration (i)-(ii), define $z^k \doteq x^k + b^k$ for all k. Under (24), (25) permits us to rewrite (i) as

$$y^{k+1} = (I + cM)^{-1}\left(r_{\mathcal{B}}(z^k) - cq\right) .$$

In terms of $z^{k+1} = x^{k+1} + b^{k+1}$, where $(x^{k+1}, b^{k+1}) \in T_2 = N_{\mathcal{B}}$, (ii) is

$$z^{k+1} = 2\lambda_k y^{k+1} + (1 - 2\lambda_k)\left(z^k\right)_{\mathcal{B}} + \left(z^k - \left(z^k\right)_{\mathcal{B}}\right) .$$

Using (25) and some simplification, one obtains

$$z^{k+1} = z^k + 2\lambda_k\left[(I + cM)^{-1}\left(r_{\mathcal{B}}(z^k) - cq\right) - \left(z^k\right)_{\mathcal{B}}\right] . \tag{26}$$

The following result is obtained by applying Proposition 6 and remarking that $\operatorname{avi}(M, q, \mathcal{B})$ must have a solution if M is positive semidefinite and $\operatorname{feas}(M, q, \mathcal{B}) \neq \emptyset$ [2].

Proposition 7 *Consider the problem* $\mathrm{avi}(M, q, \mathcal{B})$ *of (1), where M is positive semidefinite, $q \in \Re^n$ and $\mathcal{B}$ is a closed convex set, and assume* $\mathrm{feas}(M, q, \mathcal{B}) \neq \emptyset$. *Take any scalar $c > 0$ and sequence $\{\lambda_k\} \subset \Re^n$ with $0 < \inf\{\lambda_k\}_{k=0}^{\infty} \leq \sup\{\lambda_k\}_{k=0}^{\infty} < 1$. Then any sequence $\{z^k\} \subset \Re^n$ conforming to (26) will converge to some z^* such that $x^* \doteq (z^*)_{\mathcal{B}}$ solves* $\mathrm{avi}(M, q, \mathcal{B})$, *and $(-1/c)(z^* - x^*) = Mx^* + q$. If $\sup\{\lambda_k\}_{k=0}^{\infty} = 1$, the same convergence is guaranteed provided M is positive definite.*

We further note that by letting $\beta_k > 0$ in Proposition 6, the iteration (26) will still converge even if $(I + cM)^{-1}(r_{\mathcal{B}}(z^k) - cq)$ is computed inexactly, provided the accuracy improves sufficiently with k.

Consider briefly the special case of the LCP (3), where $\mathcal{B} = \Re^n_+$. It is clear that $r_{\mathcal{B}}(z) = |z|$, the component-wise absolute value of z. Setting $\lambda_k \equiv 1/2$, we obtain the very simple method

$$z^{k+1} = (I + cM)^{-1}\left(\left|z^k\right| - cq\right) + \left(z^k\right)^- \quad , \tag{27}$$

where $\left(z^k\right)^-$ is the component-wise negative part of z^k. Proposition 7 shows this recursion converges to some $z^* = x^* - c(Mx^* + q)$, where x^* solves (3), subject only to positivity of c and positive semidefiniteness of M. From such a z^*, one can compute x^* as the component-wise positive part of x^*, $x^* = (z^*)^+$.

2.2 Matrix Splitting Schemes

Another possibility is to take two $n \times n$ positive semidefinite matrices M_1 and M_2 with $M_1 + M_2 = M$, and two vectors $q^1, q^2 \in \Re^n$ such that $q^1 + q^2 = q$, and let

$$\begin{array}{rcll} T_1(x) & = & \{M_1 x + q^1\} + N_{\mathcal{B}}(x) & \forall x \in \Re^n \\ T_2(x) & = & \{M_2 x + q^2\} & \forall x \in \Re^n \, . \end{array} \tag{28}$$

Then we have

$$\begin{array}{rcl} (T_1 + T_2)(x) \ni 0 & \Leftrightarrow & \left\{M_1 x + q^1\right\} + \left\{M_2 x + q^2\right\} + N_{\mathcal{B}}(x) \ni 0 \\ & \Leftrightarrow & \{Mx + q\} + N_{\mathcal{B}}(x) \ni 0 \, , \end{array}$$

and $(T_1 + T_2)(x) \ni 0$ if and only if x solves (23), and hence $\mathrm{avi}(M, q, \mathcal{B})$. T_1 and T_2 are both maximal monotone by the same arguments used above.

However, we will consider a generalization of (28) with the same change of variables applied simultaneously to T_1 and T_2:

$$\begin{array}{rcll} T_1(\tilde{x}) & = & \{Q^\top M_1 Q\tilde{x} + Q^\top q^1\} + Q^\top N_{\mathcal{B}}(Q\tilde{x}) & \forall \tilde{x} \in \Re^n \\ T_2(\tilde{x}) & = & \{Q^\top M_2 Q\tilde{x} + Q^\top q^2\} & \forall \tilde{x} \in \Re^n \quad , \end{array} \tag{29}$$

where Q is an arbitrary nonsingular matrix. In this case, $\tilde{x}$ is a zero of $T_1 + T_2$ if and only if $x = Q\tilde{x}$ solves $\mathrm{avi}(M, q, \mathcal{B})$. The identifications (29) will give different algorithms than those arising from (24) unless $M_1 = 0$ and $Q = I$.

Now consider applying Proposition 6 to the identifications (29). We start from (i)-(ii), but in terms of variables $(\tilde{x}^k, b^k) \in T_2$ and $(\tilde{y}^k, a^k) \in T_1$. The form of T_2 gives $b^k = Q^\top(M_2 Q\tilde{x}^k + q^2)$. Thus, (i) involves solving for $\tilde{y}^{k+1}$ in the inclusion

$$\tilde{y}^{k+1} + Q^\top\left(M_1 Q\tilde{y}^{k+1} + q^1 + N_{\mathcal{B}}(Q\tilde{y}^{k+1})\right) \ni \tilde{x}^k - Q^\top\left(M_2 Q\tilde{x}^k + q^2\right) \quad .$$

Premultiplying by $Q^{-\top}$ and substituting $x^k = Q\tilde{x}^k$ and $y^k = Q\tilde{y}^k$ for all k, we then have

$$\begin{array}{cc} & Q^{-\top}Q^{-1}y^{k+1} + M_1 y^{k+1} + q^1 + N_{\mathcal{B}}(y^{k+1}) \ni Q^{-\top}Q^{-1}x^k - M_2 x^k - q^2 \\ \Leftrightarrow & Hy^{k+1} + M_1 y^{k+1} + q + (M_2 - H)x^k + N_{\mathcal{B}}(y^{k+1}) \ni 0 \\ \Leftrightarrow & y^{k+1} \in \mathrm{sol}\Big(H + M_1, q + (M_2 - H)x^k, \mathcal{B}\Big) \quad , \end{array}$$

where $H \doteq Q^{-\top}Q^{-1}$. Since H is positive definite and M_1 is positive semidefinite, $H+M_1$ is positive definite, and the "sol" denotes a unique point [34, Theorem 4.3] (this result also confirms that y^{k+1} must exist).

After a similar change of variables, (ii) reduces to

$$x^{k+1} = (H + M_2)^{-1}\left(H\left(2\lambda_k y^{k+1} + (1 - 2\lambda_k)x^k\right) + M_2 x^k\right) .$$

Note that if M_2 is positive definite, then T_2 will be strongly monotone and Lipschitz (the same cannot be said about M_1 and T_1 because $N_{\mathcal{B}}$ is not Lipschitz). The following result is now immediate from Proposition 6.

Proposition 8 *Let M_1 and M_2 be $n \times n$ real positive semidefinite matrices, define $M \doteq M_1 + M_2$, and take any $q \in \Re^n$. Let H any symmetric positive definite matrix and suppose $\{(x^k, y^k)\} \in \Re^n \times \Re^n$ conforms to the recursions*

$$y^{k+1} = \mathrm{sol}\Big(H + M_1, q + (M_2 - H)\,x^k, \mathcal{B}\Big) \tag{30}$$

$$x^{k+1} = (H + M_2)^{-1}\left(H\left(2\lambda_k y^{k+1} + (1 - 2\lambda_k)x^k\right) + M_2 x^k\right) . \tag{31}$$

If $\mathrm{feas}(M, q, \mathcal{B}) \neq \emptyset$ *and* $0 < \inf\{\lambda_k\}_{k=0}^{\infty} \leq \sup\{\lambda_k\}_{k=0}^{\infty} < 1$*, then $\{x^k\}$ converges to a solution of* $\mathrm{avi}(M, q, \mathcal{B})$*. The same convergence is also guaranteed if* $\mathrm{feas}(M, q, \mathcal{B}) \neq \emptyset$*,* $0 < \inf\{\lambda_k\}_{k=0}^{\infty} \leq \sup\{\lambda_k\}_{k=0}^{\infty} \leq 1$*, and M_2 is positive definite.*

The nonexpansiveness (and hence Lipschitz continuity) of the operator $(I + T_1)^{-1}$ can be used to show that the auxiliary sequence $\{y^k\}$ converges to the same limit x^* as $\{x^k\}$.

Now suppose we are given a positive semidefinite M and consider any splitting of M into $M = B + C$, where B is positive definite, but $C = M - B$ is arbitrary. Suppose that we set

$$H = \mathrm{sym}(B)\ ,\ M_1 = \mathrm{skew}(B)\ ,\ M_2 = M - \mathrm{skew}(B)\ , \tag{32}$$

where $\mathrm{sym}(B) \doteq \frac{1}{2}(B + B^\top)$ and $\mathrm{skew}(B) \doteq B - \mathrm{sym}(B) = \frac{1}{2}(B - B^\top)$ denote the symmetric and skew-symmetric parts of B, respectively.

Under (32), H is symmetric positive definite, and both M_1 and M_2 are positive semidefinite. Applying (30)-(31) and simplifying, we immediately obtain

$$y^{k+1} = \mathrm{sol}\Big(B, Cx^k + q, \mathcal{B}\Big) \tag{33}$$

$$x^{k+1} = (2\,\mathrm{sym}(B) + C)^{-1}\left(2\,\mathrm{sym}(B)\left(\lambda_k y^{k+1} + (1 - \lambda_k)x^k\right) + Cx^k\right) . \tag{34}$$

These recursions cause $\{x^k\}$ to converge to a solution of (1) subject only to positive definiteness of B, positive semidefiniteness of $M = B + C$, and $\{\lambda_k\}$ being bounded away from 0 and 1. The method may be construed as standard matrix splitting (see *e.g.* [26, 27, 28]), with the computation (34) replacing the usual under-relaxation step. Proposition 8 also guarantees convergence when $\{\lambda_k\}$ is allowed to approach 1, so long as M, and hence $M_2 = M - \mathrm{skew}(B)$, is positive definite.

To close this section, we note that the calculations in (30)-(31) or (33)-(34) may be performed approximately in the manner described in Proposition 6. Also, although it is beyond the scope of this paper, the techniques employed here may also be applied to produce splitting methods for *nonlinear* monotone variational inequalities.

3 Parallel Application to Optimal Control

We now present a case study indicating that splitting methods can be an effective computational tool for difficult MAVI problems. In particular, one can use splitting to dissect a problem into comparatively simple substructures that can be solved using highly parallel techniques. We draw an example of this kind of decomposition from the field of discrete-time optimal control. Our formulation is based on the notion of extended linear-quadratic programming as introduced by Rockafellar [40, 41].

3.1 Discrete-Time Optimal Control as an Extended Linear-Quadratic Problem

Many optimal control problems have essentially linear dynamics that evolve over a fixed time interval $[\tau_L, \tau_R]$ according to an underlying differential equation

$$\frac{dw}{d\tau}(\tau) = \tilde{A}(\tau)w(\tau) + \tilde{B}(\tau)u(\tau) + \tilde{b}(\tau) \ \ \forall \tau \in (\tau_L, \tau_R) \quad , \tag{35}$$

with various auxiliary conditions specified at each instant in the time interval. The function $w : [\tau_L, \tau_R] \to \Re^s$ represents the state of the system at any time instant, while

$$u(\tau) \in \mathcal{U}(\tau) \subseteq \Re^h \ \ \forall \tau \in (\tau_L, \tau_R) \tag{36}$$

are the control variables at each time instant, and

$$u^L \in \mathcal{U}_L \subseteq \Re^{h_L} \tag{37}$$

represents an initial control. Table 1 summarizes all the relevant problem data. The matrices and sets parametrized by τ are usually assumed to vary continuously with τ. Initial state conditions can be specified using

$$w(\tau_L) = B^L u^L + b^L \quad ; \tag{38}$$

a fixed initial state can be generated by choosing $B^L = 0$.

The beauty of the formulation advocated by Rockafellar is its ability to easily model constraints on the states and the controls at both intermediate and terminal times. This capability stems from *monitoring functions* $\mu_{\mathcal{V}Q}$ defined via

$$\mu_{\mathcal{V}Q}(z) \doteq \sup_{v \in \mathcal{V}} \left\{ \langle v, z \rangle - \tfrac{1}{2} \langle v, Qv \rangle \right\} \quad .$$

Here $\mathcal{V}$ is a subset of $\Re^m$ and Q is a positive semidefinite $m \times m$ matrix. The function $\mu_{\mathcal{V}Q}$ acts as a (linear-quadratic) penalty function and is allowed to take the value $+\infty$. Many different penalizations of the state and control "constraints" can be added using particular choices of $\mathcal{V}$ and Q. For example, if $\mathcal{V} = \Re^m_+$ and $Q = 0$, then $\mu_{\mathcal{V}Q}(z) = 0$ if $z \leq 0$, and $+\infty$ otherwise, effectively generating inequality constraints. [40] gives a variety of choices of $\mathcal{V}$ and Q, showing how to model terminal state conditions and linear-quadratic regulator problems.

The general problem formulation is to choose $u(\tau)$ and u^L to minimize the functional

$$\begin{aligned} \mathcal{F}(u^L, u) \ \doteq \ & \int_{\tau_L}^{\tau_R} \tilde{p}(\tau)u(\tau) + \tfrac{1}{2}\left\langle u(\tau), \tilde{P}(\tau)u(\tau)\right\rangle - \langle \tilde{c}(\tau), w(\tau)\rangle \, d\tau \\ & + \left\langle (p^L), u^L \right\rangle + \tfrac{1}{2}\left\langle (u^L), P^L u^L \right\rangle - \left\langle (c^R), w(\tau_R) \right\rangle \\ & + \int_{\tau_L}^{\tau_R} \mu_{\mathcal{V}(\tau)\tilde{Q}(\tau)}\left(\tilde{r}(\tau) - \tilde{C}(\tau)w(\tau) - \tilde{D}(\tau)u(\tau)\right) \, d\tau \\ & + \mu_{\mathcal{V}_R Q^R}\left(r^R - C^R w(\tau_R)\right) \end{aligned} \tag{39}$$

Data	**Type of Data**
s, h, h_L, m, m_R	Positive integers
$[\tau_L, \tau_R]$	A closed real interval
$\mathcal{U}_L \subseteq \Re^{h_L}, \mathcal{V}_R \subseteq \Re^{m_R}$	Closed convex sets
$\mathcal{U}(\tau) \subseteq \Re^h, \mathcal{V}(\tau) \subseteq \Re^m$	Closed convex set functions of $\tau \in [\tau_L, \tau_R]$
$p_L \in \Re^{h_L}, b^L, c^R \in \Re^s, r^R \in \Re^{m_R}$	Real vectors
$P^L \in \Re^{h_L \times h_L}, Q^R \in \Re^{m_R \times m_R}$	Positive semidefinite matrices
$B^L \in \Re^{s \times h_L}, C^R \in \Re^{m_R \times s}$	Arbitrary real matrices
$\tilde{p}(\tau) \in \Re^h, \tilde{c}(\tau), \tilde{b}(\tau) \in \Re^s, \tilde{r}(\tau) \in \Re^m$	Real vector functions of $\tau \in [\tau_L, \tau_R]$
$\tilde{P}(\tau) \in \Re^{h \times h}, \tilde{Q}(\tau) \in \Re^{m \times m}$	Positive semidefinite matrix functions of $\tau \in [\tau_L, \tau_R]$
$\tilde{A}(\tau) \in \Re^{s \times s}, \tilde{B}(\tau) \in \Re^{s \times m}$, $\tilde{C}(\tau) \in \Re^{m \times s}, \tilde{D}(\tau) \in \Re^{m \times h}$	Arbitrary real matrix functions of $\tau \in [\tau_L, \tau_R]$.

Table 1: Data describing an instance of an optimal control problem.

subject to the constraints (35)-(38).

In order to solve such a problem, we consider a discretization of (35)-(39) using N time points

$$\tau_1 = \tau_L, \ldots, \tau_t = \left(\tfrac{N-t}{N-1}\right)\tau_L + \left(\tfrac{t-1}{N-1}\right)\tau_R, \ldots, \tau_N = \tau_R \,.$$

The integrals in (39) are approximated by finite sums, and the derivative in (35) is modeled using a finite-difference formula. For further details, refer to [7, Section 3.6]. If we relabel the discretized variables using

$$\begin{array}{rclcrcl} u_{[1]} &=& u^L & \qquad & u_{[t]} &=& u(\tau_{t-1}), \quad t = 2, \ldots, N \\ w_{[t]} &=& w(\tau_t), \quad t = 1, \ldots, N-1 & & w_{[N]} &=& w^R \quad , \end{array}$$

then the resulting discretized problem is

$$\begin{array}{ll} \min & \displaystyle\sum_{t=1}^{N} \left(\left\langle p_{[t]}, u_{[t]} \right\rangle + \frac{1}{2} \left\langle u_{[t]}, P_{[t]} u_{[t]} \right\rangle - \left\langle c_{[t]}, w_{[t]} \right\rangle + \mu_{\mathcal{V}_{[t]} Q_{[t]}} (r_{[t]} - C_{[t]} w_{[t]} - D_{[t]} u_{[t+1]}) \right) \\ \text{S.T.} & w_{[t]} = A_{[t-1]} w_{[t-1]} + B_{[t]} u_{[t]} + b_{[t]} \qquad t = 1, \ldots, N \\ & u_{[t]} \in \mathcal{U}_{[t]} \qquad\qquad\qquad\qquad\qquad\;\; t = 1, \ldots, N \quad . \end{array} \tag{40}$$

Note the time shift in the definition of $w_{[t]}$ and $u_{[t]}$. $P_{[t]}$, $\mathcal{U}_{[t]}$, $Q_{[t]}$, $\mathcal{V}_{[t]}$, $A_{[t]}$, $B_{[t]}$, $C_{[t]}$, $D_{[t]}$, $p_{[t]}$, $b_{[t]}$, $r_{[t]}$, and $c_{[t]}$ are chosen to correspond to the continuous problem. For example, letting $\delta = (\tau_R - \tau_L)/(N-1)$, we have $P_{[1]} = P^L$, $\mathcal{U}_{[1]} = \mathcal{U}^L$, $P_{[t]} = \delta \tilde{P}(\tau_{t-1})$ and $\mathcal{U}_{[t]} = \mathcal{U}(\tau_{t-1})$ for $t = 2, \ldots, N$, $c_{[t]} = \delta \tilde{c}(\tau_t)$ for $t = 1, \ldots, N-1$, and $c_{[N]} = c^R$. Note that $P_{[t]}$ and $Q_{[t]}$ are square positive semidefinite matrices, while $A_{[t]}$, $B_{[t]}$, $C_{[t]}$, and $D_{[t]}$ are arbitrary, and possibly nonsquare and dense. We assume implicitly that $w_{[0]}$ and $u_{[N+1]}$ are identically zero.

This discretized problem is also an extended linear-quadratic programming problem, for which there is an elegant duality theory based on saddle-point theory for a Lagrangian

$$\begin{aligned} \mathcal{L}(u,y,v,w) \;&\doteq\; \sum_{t=1}^{N} \left\langle p_{[t]}, u_{[t]} \right\rangle + \tfrac{1}{2}\left\langle u_{[t]}, P_{[t]} u_{[t]} \right\rangle \\ &\quad - \left\langle c_{[t]}, w_{[t]} \right\rangle + \left\langle v_{[t]}, r_{[t]} - C_{[t]} w_{[t]} - D_{[t]} u_{[t+1]} \right\rangle \\ &\quad - \tfrac{1}{2}\left\langle v_{[t]}, Q_{[t]} v_{[t]} \right\rangle + \left\langle y_{[t]}, w_{[t]} - A_{[t-1]} w_{[t-1]} - B_{[t]} u_{[t]} - b_{[t]} \right\rangle \quad , \end{aligned} \tag{41}$$

where $u_{[t]} \in \mathcal{U}_{[t]}$ and $v_{[t]} \in \mathcal{V}_{[t]}$ for $t = 1, \ldots, N$.

The primal problem (40) arises from maximizing this Lagrangian over v and y; the corresponding dual problem results from minimizing the Lagrangian over u and w. Essentially, this dual problem is also a control problem where v are the (discretized) dual control variables and y are the dual state variables,

It is well-known that determining a saddle point of (41) is equivalent to solving (40) under a suitably mild qualification [40]. After some simplifications, the discretized saddle-point problem in the variables $x_{[t]} = (u_{[t]}, y_{[t]}, v_{[t]}, w_{[t]})$, $t = 1, 2, \ldots, N$ is equivalent to $\mathrm{avi}(M, q, \mathcal{B})$, with

$$M = \begin{bmatrix} M_{[1]} & -L_{[1]}^\top & & & & \\ L_{[1]} & M_{[2]} & -L_{[2]}^\top & & & \\ & L_{[2]} & M_{[3]} & -L_{[3]}^\top & & \\ & & & \ddots & & \\ & & L_{[N-2]} & M_{[N-1]} & -L_{[N-1]}^\top & \\ & & & L_{[N-1]} & M_{[N]} & \end{bmatrix}, \tag{42}$$

$$q = \begin{bmatrix} q_{[1]} & q_{[2]} & \cdots & q_{[N]} \end{bmatrix}^\top, \tag{43}$$

$$\mathcal{B} = \prod_{t=1}^{N} \left(\mathcal{U}_{[t]} \times \Re^s \times \mathcal{V}_{[t]} \times \Re^s \right) \;, \tag{44}$$

where the submatrices $M_{[t]}$, $L_{[t]}$, and $q_{[t]}$ take the respective forms

$$M_{[t]} = \begin{bmatrix} P_{[t]} & -B_{[t]}^\top & & \\ B_{[t]} & & & -I \\ & & Q_{[t]} & C_{[t]} \\ & I & -C_{[t]}^\top & \end{bmatrix}, \tag{45}$$

$$L_{[t]} = \begin{bmatrix} & -D_{[t]}^\top & \\ & & A_{[t]} \\ & & \\ & & \end{bmatrix}, \tag{46}$$

$$q_{[t]} = \begin{bmatrix} p_{[t]}^\top & b_{[t]}^\top & -r_{[t]}^\top & -c_{[t]}^\top \end{bmatrix}. \tag{47}$$

The matrices $M_{[t]}$ are square, with $M_{[1]}$ having dimension $h_L + 2s + m$, $M_{[N]}$ having dimension $h + 2s + m_R$, and the rest having dimension $h + 2s + m$.

We generated test problems having precisely this structure using techniques already developed for MCPLIB [7], which is based on the code written by Wright [50] at the University of Washington. This procedure generates affine variational inequalities of the form (42)-(47) with the property that

$\mathcal{U}_{[t]}$ and $\mathcal{V}_{[t]}$ are products of bounded closed intervals, $P_{[t]}$ and $Q_{[t]}$ are diagonal, and $\tilde{A}(\tau)$, $\tilde{B}(\tau)$, $\tilde{C}(\tau)$, $\tilde{D}(\tau)$, $\tilde{P}(\tau)$, $\tilde{Q}(\tau)$, $\mathcal{U}(\tau)$, and $\mathcal{V}(\tau)$ do not vary with τ. Although this last property makes all the matrices $M_{[t]}$ identical, except for $M_{[1]}$ and $M_{[N]}$, this characteristic does not appear to make the generated problems particularly easy to solve, and our code takes no advantage of it.

Thus, all of out test problems have the dynamical structure that is described in [43]. Only the parameters referred to within this structure have been randomized. Furthermore, the assumed structure is very general, and includes many standard problems occurring in optimal control.

3.2 Parallel Application of Splitting

Consider the application of splitting algorithms to MAVI's with the structure described in (42)-(47). If the $P_{[t]}$ and $Q_{[t]}$ are all positive definite, then algebraically eliminating the free variables yields a more compact positive definite AVI of dimension $(N-1)(h+m)+h_L+m_R$. However, the resulting matrix no longer has the block tridiagonal structure of (42), but is instead block lower triangular, and very much denser. Instead, we chose to maintain the original problem structure, and adopt the approach of (30)-(31), with $\sup\{\lambda_k\}_{k=0}^{\infty} < 1$. An approach based on forward-backward splitting is also possible [4], but is subject to relatively stringent stepsize restrictions.

Our approach is motivated by the existence of parallel solvers for block-tridiagonal systems of linear equations having structure like (42). Unfortunately, these techniques do not generalize directly to variational inequality problems, because it is no longer always possible to use one row to "eliminate" another. However, one can isolate the block tridiagonal structure of the problem by taking the approach of Proposition 8 with

$$M_1 = \begin{bmatrix} \alpha M_{[1]} & & & \\ & \alpha M_{[2]} & & \\ & & \ddots & \\ & & & \alpha M_{[N]} \end{bmatrix}, \tag{48}$$

$$M_2 = \begin{bmatrix} (1-\alpha)M_{[1]} & -L_{[1]}^{\top} & & \\ L_{[1]} & (1-\alpha)M_{[2]} & -L_{[2]}^{\top} & \\ & & \ddots & \\ & & L_{[N-1]} & (1-\alpha)M_{[N]} \end{bmatrix}, \tag{49}$$

and α being some scalar in the range $[0,1]$. If we choose H to be block diagonal conformally with M_1, then $H+M_1$ is also block diagonal. Because of the separable structure of $\mathcal{B}$ in (44), step (30) now decomposes into N independent, smaller AVI problems, each one over a box $\mathcal{B}_{[t]} \doteq \mathcal{U}_{[t]} \times \Re^s \times \mathcal{V}_{[t]} \times \Re^s$. These problems may be solved independently and in parallel. Similarly, with H block diagonal, the matrix $H+M_2$ in (31) will be block tridiagonal. Thus, (31) may be solved by block-oriented versions of the parallel cyclic reduction and substructuring methods described in [15, Section 5.4] and [16]. Briefly, block cyclic reduction involves using block Gaussian elimination to eliminate every other block of rows. The remaining, uneliminated rows form a block tridiagonal system half the size of the original one. Eliminating every other block of rows from this system and proceeding recursively, one may factor the system in $\log_2 N$ parallel steps.

When $\alpha = 1$, a much simpler procedure is possible with some further assumptions on H. In addition to H being block diagonal, that is,

$$H = \begin{bmatrix} H_{[1]} & & & \\ & H_{[2]} & & \\ & & \ddots & \\ & & & H_{[N]} \end{bmatrix}, \tag{50}$$

suppose that each $H_{[t]}$ has the block diagonal substructure

$$H_{[t]} = \begin{bmatrix} H^1_{[t]} & & & \\ & H^2_{[t]} & & \\ & & H^3_{[t]} & \\ & & & H^4_{[t]} \end{bmatrix}, \tag{51}$$

with the blocks conforming to those of $M_{[t]}$. Then $H + M_2$ takes the form

$$\left[\begin{array}{ccccccccccc}
H^1_{[1]} & & & & & & & & & & \\
& H^2_{[1]} & & & & & & & & & \\
& & H^3_{[1]} & & D_{[1]} & & & & & & \\
& & & H^4_{[1]} & & -A^\top_{[1]} & & & & & \\
& & -D^\top_{[1]} & & H^1_{[2]} & & & & & & \\
& & & A_{[1]} & & H^2_{[2]} & & & & & \\
& & & & & & H^3_{[2]} & & D_{[2]} & & \\
& & & & & & & H^4_{[2]} & & -A^\top_{[2]} & \\
& & & & & & -D^\top_{[2]} & & H^1_{[3]} & & \\
& & & & & & & A_{[2]} & & H^2_{[3]} & \\
& & & & & & & & & & \ddots
\end{array}\right].$$

This matrix is also block diagonal, and the linear system of (31) thus decomposes into many smaller, independent linear systems. Note, however, that the system's block structure does not align exactly with that of M in (42); instead, its blocks are "offset."

We now have a parallel means for implementing both steps (30) and (31) of our splitting iteration. In practice, we also need some procedure for terminating the algorithm. Given that $\mathcal{B}$ is a "box" defined by $\ell \in [-\infty, +\infty)^n$ and $u \in (-\infty, +\infty]^n$ as in (2), we may define, for any $x, g \in \Re^n$,

$$\gamma_i(x_i, g_i; \ell_i, u_i) \doteq \max\left\{0, \ell_i - x_i, x_i - u_i, \min\left\{x_i - \ell_i, g_i\right\}, \min\left\{u_i - x_i, -g_i\right\}\right\} \tag{52}$$

$$\Gamma(x, g; \ell, u) \doteq \max_{i=1,\ldots,n} \left\{\gamma_i(x_i, g_i; \ell_i, u_i)\right\}. \tag{53}$$

Then the problem $\text{avi}(M, q, \mathcal{B})$ given in (1) is equivalent to finding some x such that $\Gamma(x, Mx + q; l, u) = 0$. In practice, we will settle for $\Gamma(x, Mx + q; l, u) \leq \epsilon$, where ϵ is some small tolerance; a simple way to terminate (30)-(31) is to periodically compute $\gamma^k \doteq \Gamma(x^k, Mx^k + q; l, u)$, and terminate if $\gamma^k \leq \epsilon$.

However, we advocate a somewhat more complicated procedure designed to detect exact solutions to (1) that might lie near the current iterate. Define $\theta^k \doteq q + (M_2 - H)x^k$ and suppose we solve each subproblem $\text{avi}(H_{[t]} + \alpha M_{[t]}, \theta^k_{[t]}, \mathcal{B}_{[t]})$ making up (30) by a standard pivotal method [5, 18]. Then for each of these subproblems there will be a final complementary basis $\mathcal{Y}^k_{[t]}$. Consider their concatenation $\mathcal{Y}^k \doteq (\mathcal{Y}^k_{[1]}, \mathcal{Y}^k_{[2]}, \ldots, \mathcal{Y}^k_{[N]})$. Eventually, as $\{y^k\}$ and $\{x^k\}$ converge to some solution x^* of $\text{avi}(M, q, \mathcal{B})$, $\mathcal{Y}^k$ should stabilize at some complementary basis corresponding to x^*. We propose to take advantage of the possibility that this basis might be encountered long before $\gamma_k \le \epsilon$.

Suppose $\mathcal{Y}^k$ does not change over χ consecutive iterations. Then by setting the nonbasic variables at their corresponding bounds and solving for the remaining basic variables, we may be able to"jump" to some solution x^*. We use following procedure to attempt such "jumps": let $\overline{y}^{k+1} \in \Re^n$ be defined by

$$\overline{y}_i^{k+1} = \begin{cases} y_i^{k+1} & y_i^{k+1} \text{ nonbasic in } \mathcal{Y}^k \\ 0 & y_i^{k+1} \text{ basic in } \mathcal{Y}^k \, , \end{cases} \tag{54}$$

and let $\overline{q}^{k+1} \doteq q + M\overline{y}^{k+1}$. Define $\overline{M}^k$ to be M with each nonbasic column in $\mathcal{Y}^k$ replaced by the corresponding canonical unit vector. Now, we solve the system $\overline{M}^k \tilde{y}^{k+1} = -\overline{q}^{k+1}$ for $\tilde{y}^{k+1}$. This system is block tridiagonal, so it is again amenable to the same parallel techniques as (31). Let $\hat{y}^{k+1} \in \Re^n$ be defined by

$$\hat{y}_i^{k+1} = \begin{cases} y_i^{k+1} & y_i^{k+1} \text{ nonbasic in } \mathcal{Y}^k \\ \tilde{y}_i^{k+1} & y_i^{k+1} \text{ basic in } \mathcal{Y}^k \, . \end{cases} \tag{55}$$

It is a simple matter to compute $\hat{\gamma}_k \doteq \Gamma(\hat{y}^{k+1}, M\hat{y}^{k+1} + q; l, u)$ from y^{k+1} and $\tilde{y}^{k+1}$. If $\hat{\gamma}_k \le \epsilon$, we terminate with the solution $\hat{y}^{k+1}$.

As a heuristic acceleration technique, we propose checking whether $\hat{\gamma}^{k+1} < \sigma\gamma_k$, where $\sigma \in [0, 1)$ is a parameter. If so, we redefine $y^{k+1} \leftarrow \hat{y}^{k+1}$ before proceeding to step (31).

We are now ready to state a complete algorithm for an AVI with the structure (42)-(47):

0. If $\alpha < 1$, factor the block tridiagonal system $H + M_2$. Set $k = 0$, choose some arbitrary starting point $x^0 \in \Re^n$, and compute $\eta^0 \doteq M_2 x^0$.

1. Compute $\theta^k = q + \eta^k - Hx^k$. Then, for all $t = 1, \ldots, N$, compute
$$y_{[t]}^{k+1} = \text{sol}\Big(H_{[t]} + \alpha M_{[t]}, \theta^k_{[t]}, \mathcal{B}_{[t]}\Big) \tag{56}$$
using a standard pivoting algorithm. If $k > 0$, use the prior complementary basis $\mathcal{Y}^{k-1}_{[t]}$ as a starting point. When done, save the final complementary basis $\mathcal{Y}^k_{[t]}$.

2. If no $\mathcal{Y}^k_{[t]}$ has changed in the last χ iterations, go to step 7.

3. Compute
$$\phi^k \doteq H\left(2\lambda_k y^{k+1} + (1 - 2\lambda_k)x^k\right) + M_2 x^k = H\left(2\lambda_k y^{k+1} + (1 - 2\lambda_k)x^k\right) + \eta^k \, . \tag{57}$$
Then solve the block tridiagonal system
$$(H + M_2)x^{k+1} = \phi^k \, . \tag{58}$$

4. Compute $\eta^{k+1} \doteq M_2 x^{k+1}$.

5. Compute $g^{k+1} \doteq Mx^{k+1} + q = \eta^{k+1} + M_1 x^{k+1} + q$. Then find $\gamma_{k+1} = \Gamma(x^{k+1}, g^{k+1}; l, u)$.

6. If $\gamma_{k+1} \le \epsilon$, terminate with the solution x^{k+1}. Otherwise, set $k \leftarrow k + 1$ and go to step 1.

7. Compute $\overline{q}^{k+1} = q + M\overline{y}^{k+1}$. Form $\overline{M}^k$, solve the block tridiagonal system $\overline{M}^k \tilde{y}^{k+1} = -\overline{q}^{k+1}$, and form $\hat{y}^{k+1}$ using (55).

8. Compute $\hat{\gamma}_k = \Gamma(\hat{y}^{k+1}, M\hat{y}^{k+1} + q; l, u)$. If $\hat{\gamma}_k \le \epsilon$, halt with the solution $\hat{y}^{k+1}$. Otherwise, if $\hat{\gamma}^k < \sigma\gamma_k$, overwrite $y^{k+1} \leftarrow \hat{y}^{k+1}$. Proceed to step 3.

Now suppose that we have a computer system consisting of $V \le N$ processors; if we have more than N processors available, we only use the first N. We allocate the time steps $t = 1, \ldots, N$ to the processors in contiguous groups of roughly N/V. Each processor stores all data associated with the rows of M and q corresponding to its allocated time steps. All working vectors in the algorithm are partitioned similarly.

Given this simple data distribution, we now describe, step by step, how to construct a data-parallel implementation of the algorithm 0-8:

0. If $\alpha < 1$, we must factor $H + M_2$, which can be done using standard parallel techniques (*e.g.* block versions of the cyclic reduction method of [16]). Computing $\eta^0 \doteq M_2 x^0$ requires an exchange of data between processors holding adjacent groups of time steps.

1. Because H is block-diagonal, the computation of θ^k decomposes by the time step t. Similarly, the computation of y^{k+1} via (56) decomposes into a collection of N independent problems, one for each time step, each of which may be solved by local application of a standard pivoting algorithm. Thus, no interprocessor communication is needed in this step.

2. Here, we must determine if any $\mathcal{Y}^k_{[t]}$ has changed in the last χ iterations. Each processor can locally determine whether any of its $\mathcal{Y}^k_{[t]}$ have so changed. Then, the processors do a global "or" reduction/broadcast operation to globally determine if any $\mathcal{Y}^k_{[t]}$ has changed.

3. Since $\eta^k = M_2 x^k$ has already been computed, (57) decomposes by timestep and requires only local computation. If $\alpha < 1$, solving (58) is a parallel back-solve based on the factorization already computed in step 0. If $\alpha = 1$, the system can be solved using only local computation and data exchange between consecutive processors.

4. Computing η^{k+1} requires data exchange between consecutive processors.

5. Finding g^{k+1} decomposes by time step, and requires only local computation. To find γ_{k+1}, each processor first computes and finds the maximum of the values $\gamma_i(x_i^{k+1}, g_i^{k+1}; \ell_i, u_i)$ (see (52)) for the indices i that it "owns". Then a single scalar interprocessor "reduction" operation finds and broadcasts the global maximum.

6. Since γ_{k+1} has just been broadcast in the previous step, no communication is required.

7. Computing $\overline{q}^{k+1}$ again requires an exchange of data between consecutive processors. Forming $\overline{M}^k$ requires a similar data exchange. Solving $\overline{M}^k \tilde{y}^{k+1} = -\overline{q}^{k+1}$ requires a factor and back-solve using standard parallel block cyclic reduction techniques. Finally, forming $\hat{y}^{k+1}$ using (55) requires only local computation.

8. Computing $\hat{\gamma}_k$ requires some local computation and a single reduction/broadcast operation, as in step 5. After the broadcast of $\hat{\gamma}_k$, no further communication is needed.

The communication requirements of the algorithm are those implicit in the block tridiagonal factor and solve operations (steps 0, 3, and 7), along with global scalar reduction (steps 2, 5, and 8) and simple one-dimensional "shift" operations (steps 0, 3, 4, and 7) for data exchange between consecutive processors. The computation and communication requirements of the algorithm are highly regular, with the possible exception of finding $y^{k+1}_{[t]}$ in step 1.

3.3 Implementation for the CM–5 Family

We implemented the algorithm on the CM–5 family of parallel computers [46]. The modified "fat tree" communication topology of the CM–5 is well-suited to parallel tridiagonal factorization, shift operations, and global reductions.

Due to the algorithm's computational and communication regularity, we chose to implement it using the global data-parallel CM Fortran (CMF) language [47] and the CMSSL collection of numerical/scientific subroutines [48], which already contains sophisticated block-tridiagonal routines based on [16]. Under the global CMF/CMSSL programming environment, the CM–5 functions essentially synchronously, like a giant array processor or SIMD computer [13]. The elemental processing units are not the "processing nodes", but the individual vector arithmetic units (VU's). Each processing node has four of these vector units. The vector units function synchronously, each performing similar operations on different data. The environment is "tuned" principally for speed of operations on long vectors.

CMF and CMSSL do not support "ragged arrays" in which the valid extent of one subscript depends on the value of another. This restriction limits our implementation to the case that all the $M_{[t]}$ have the same dimension, namely $h_L = h$ and $m_R = m$. Thus the dimension of the square matrix M simplifies to $n = N(2s + h + m)$.

The implementation also assumes that the matrices $P_{[t]}$ and $Q_{[t]}$ are diagonal, as are the scaling/stepsize matrices $H^1_{[t]}, \ldots, H^4_{[t]}$. These additional restrictions are inessential, and could be removed very easily. $A_{[t]}$, $B_{[t]}$, $C_{[t]}$, and $D_{[t]}$ are represented as dense matrices.

CMF and the CMSSL made most of the implementation straightforward. The CMSSL already contained the necessary block tridiagonal routines, and most of the local calculations were easily written in CMF or were simple applications of the CMSSL's "multiple instance" dense linear algebra functions. CMF's ANY, MAXVAL and EOSHIFT intrinsic functions provided the required global reductions and data exchanges between consecutive processors.

The main difficulty was in solving the local subproblems $\mathrm{avi}(H_{[t]} + \alpha M_{[t]}, \theta_{[t]}, \mathcal{B}_{[t]})$ in step 1. Our approach was to first algebraically eliminate each subproblem's unbounded variables, transforming it into a fully dense AVI of dimension $h+m$ on a bounded box $\mathcal{U}_{[t]} \times \mathcal{V}_{[t]}$. To this collection of problems, we then applied our own special, multiple-instance, synchronous implementation of Lemke's algorithm, written in CMF, with calls to the CMSSL. Essentially, we synchronously perform Lemke pivots on every subproblem instance $t = 1, \ldots, N$. Instances that have already located a complementary solution perform trivial basis-preserving pivots until all instances have reached termination. We use explicit representations of the basis inverses, with outer-product updates and periodic reinversion.

This approach was the easiest available to us, but has two potential drawbacks. First, after each pivot, the code must perform a fundamentally unnecessary global communication operation to determine whether all instances have terminated. Fortunately, these operations are extremely efficient on the CM–5. Second, when $N > V$, the algorithm may perform a large number of unnecessary flops, since each processor pivots on *all* its instances at each Lemke iteration, as opposed to only those that have not terminated. In a less inherently synchronous programming environment, both these inefficiencies could be removed; however, implementing the rest of the algorithm might prove considerably more complicated.

3.4 Computational Results on CM–5E Systems

For purposes of comparison, we attempted to solve the problems using standard serial codes for affine variational inequalities on a SPARCStation 10/51 workstation with 32 megabytes of RAM.

The serial codes consisted of PATH [8], which implements a generalization of the SQP method with a piecewise-linear path search, SMOOTH [3], a differentiable approximation method, and MILES [45], which implements a standard pivotal algorithm of the Lemke class. All the above codes are designed for nonlinear box-constrained variational inequalities, but can be applied to linear problems. By reformulating the test problems as quadratic programs [42], we also attempted to solve some of the problems with the generalized reduced gradient nonlinear programming code CONOPT [10]. The generated quadratic programs are convex, but they are guaranteed *not* to be strictly convex.

Other techniques for the parallel solution of ELQP problems arising in optimal control have been described in the literature. Wright [51] develops an interior point SQP approach with special adaptation of the (banded) linear algebra for solving the generated subproblems. Pantoja and Mayne [29] also use an SQP approach, but exploit the structure at a higher level. Both of these techniques are essentially comparable to the PATH method, but with special implementation to exploit the problem structure. None of these authors report results on the structured ELQP's outlined in Section 3.1.

Zhu and Rockafellar [52, 53] consider the problem as an ELQP and apply forward-backward splitting techniques exploiting the underlying duality structure. Although they report results for problems similar to the well-conditioned ones that we consider, their implementation is serial. No parallel implementations of the above methods available for the CM-5, so we report only representative serial times for those codes that were accessible within GAMS.

We ran the parallel code on three different CM–5E configurations, with a total of 16, 128, and 256 vector units, respectively. The CM–5E is a variant of the basic CM–5 in which each processing node consists of a 40-MHz SPARC–10 and four 40-MHz vector units. In all cases, each vector unit had access to 32 megabytes of local RAM.

We created two sets of test problems. To conform with the assumptions of our CM–5 implementation, we restricted the MCPLIB procedure (as described in Section 3.1) to the case $h_L = h$ and $m_R = m$ and limited the scope of our experiments by considering only the case $h = m = s$. The dimension of the problems simplifies to $n = 4Ns$ and within each set of problems, the data is parameterized by N and s. The GAMS source file that was used to generate the test problems is available by anonymous ftp from `ftp://ftp.cs.wisc.edu/math-prog/mcplib/gams/opt_cont.gms`.

In the first set, the condition numbers of the matrices $P_{[t]}$ and $Q_{[t]}$ are in the range of 2 to 3. We generated such problems with $s = 8$, 16, 24, 32, and 40. In the second set, the condition numbers of $P_{[t]}$ and $Q_{[t]}$ were much higher, in the range of 10^4 to 10^5. For these more ill-conditioned problems, we generated instances with $s = 8$, 16, and 24. For both sets, we considered all values of N between 16 and 1024, stepping by powers of 2. For a few problems, we also tried $N = 2048$.

After some careful but far from exhaustive experimentation on a subset of test problems, we arrived at the following parameter settings, where $H = \nu I$:

$$\begin{array}{rclrcl} \nu & = & \max\{8/N, 0.025\} & \epsilon & = & 10^{-7} \\ \lambda_k & \equiv & 0.9 & \sigma & = & 0.9 \\ \alpha & = & 0.05 & \chi & = & 50\,. \end{array} \tag{59}$$

Only ν (and hence H) varies with the problem to be solved. Since M_2 is not positive definite, we were constrained to keep λ_k bounded away from 1. Thus, the algorithm implemented is a form of "Douglas-Rachford" (DR) splitting as described in [12].

We set χ to be fairly large because the effort involved in factoring the block tridiagonal matrix $\overline{M}^k$ is considerable, and in particular much greater than that needed in the block tridiagonal backsolves of step 3. In practice, the algorithm always terminates in step 8. There were typically between 1 and 4 heuristic "jumps" (when the test $\hat{\gamma}_k \leq \sigma\gamma_k$ was passed), and they tended to be concentrated towards the end of each run.

For well-conditioned problems with small s, $\alpha = 1$ worked very well, cutting the time per iteration approximately in half without much penalty in iteration count. For more difficult problems, however, the algorithm is very sensitive to the value of α. Values of α near 1 inflated iteration counts by an order of magnitude or more; in general, it seemed important to use a value of α at or near 0.

The sensitivity of the algorithm to the other parameters is much milder. Setting ν to within a factor of 4 or so of the value specified in (59) resulted in only modest run time variations. Similarly, sensitivity to λ_k is limited as long as it is not close to 0, although values near 1 do tend to work best. Sensitivity to the termination tolerance ϵ is minimal unless the parameter is very large, since the algorithm always terminates with a step 8 "jump" to a solution that is accurate to at least 10 digits of precision.

Table 2 gives run times for the well-conditioned test problems. The solutions obtained by all our methods agreed to seven significant digits. In Table 2, "*Memory*" means that a given problem would not fit in physical memory, and "*Time*" indicates that the specified code could not solve the problem within 14,400 seconds (4 hours). The results are also plotted graphically in Figures 1 through 5. It should be noted that the per-timestep memory requirements of the CM–5 implementation are considerably higher than the serial implementations'. This disparity is partly due to the large temporary data structures allocated by CMSSL, and partly to our decision to use dense linear algebra in our CM–5 Lemke subroutine. The latter choice made implementation considerably easier, and probably did not require any sacrifice in speed, due to the CM–5's preference for long vector operations.

Of the serial codes, PATH appears to be the most efficient for these problems, followed (somewhat distantly) by SMOOTH. CONOPT and MILES could only solve the easiest instances. However, the parallel splitting implementation is markedly faster than any of the serial codes for almost all the problems. The exceptions are that PATH is competitive with the parallel code when $N = 16$, and also for the $s = 8$ problems that have $N \leq 256$. Depending on s, the parallel code could solve problems 2 to 8 times larger than PATH.

Comparison of the timings for the 16, 128, and 256 vector unit configurations illustrate how the splitting approach is able to take advantage of parallelism. Our code can use at most N vector units, even if more are available, so the 128- and 256-VU timings are essentially the same for $N \leq 128$. In fact, the 128-VU system gives slightly better performance in these cases, because it is controlled by a SPARC–10 "front end" processor, whereas other systems are controlled by SPARC–2 front ends. For $N = 16$, all the CM–5E configurations yield similar timings, except for the effects caused by front end performance.

Table 3 gives run times for the more ill-conditioned problems. The format is the same as for Table 2, except that there are some "*Failure*" entries in the CONOPT column. These entries indicate that, once the problems were converted to quadratic programs, CONOPT rejected them as having no feasible solution. Figures 6 through 8 show the same data graphically. The relative performance of the various codes remains quite similar to the well-conditioned problems, except that parallel implementation's advantage over PATH is even more dramatic for $s \geq 16$. For the same values of s, N, and number of vector units V, the splitting method does take longer to solve the ill-conditioned problems than the well-conditioned ones, apparently due to a slowing in the convergence rate of the underlying splitting procedure. There was an increase in the number of splitting iterations, mitigated by decrease in the number of Lemke pivots per iteration.

The tuning of the CM–5/CMF/CMSSL environment for long vector operations makes it difficult

		Serial Codes (SPARCStation 10/51)				Parallel Splitting (CM–5E)		
s	N	PATH	SMOOTH	CONOPT	MILES	16 VU	128 VU	256 VU
8	16	0.91	7.57	5.85	7.03	9.74	6.40	13.00
8	32	2.04	16.33	37.76	36.47	8.47	4.11	6.21
8	64	4.08	40.07	180.45	292.6	7.90	3.98	7.75
8	128	8.47	134.14	2688.14	1486	11.45	5.04	6.87
8	256	19.04	217.74	*Time*	*Time*	22.94	9.63	11.66
8	512	35.93	*Memory*			7.30	2.58	3.06
8	1024	93.56				12.74	3.30	3.42
8	2048	*Memory*				26.15	4.95	4.10
16	16	11.56	44.73	33.65	42.44	11.31	8.23	12.75
16	32	32.75	133.49	176.57	352.5	13.07	7.69	9.10
16	64	65.29	191.31	2717.86	3042.00	18.70	9.58	10.97
16	128	168.84	649.61	*Time*	*Time*	25.64	10.26	12.37
16	256	455.04	*Memory*			65.93	19.53	17.70
16	512	1005.45				80.95	19.09	18.48
16	1024	*Memory*				*Memory*	49.37	35.69
16	2048						100.33	63.12
24	16	47.88	126.05			18.81	14.16	19.64
24	32	152.5	388.12			24.62	15.5	17.23
24	64	418.2	1029.32			85.57	25.93	25.14
24	128	1169.46	4765.17			102.08	34.25	40.72
24	256	2814.20	*Memory*			292.85	72.83	61.51
24	512	*Memory*				*Memory*	167.26	131.27
24	1024						1042.09	621.88
32	16	54.79	280.68			58.67	45.54	47.10
32	32	178.85	989.64			81.37	52.74	59.04
32	64	536.18	2815.9			104.95	48.61	53.90
32	128	1208.22	*Memory*			455.56	114.12	129.08
32	256	*Memory*				*Memory*	320.96	262.7
32	512						822.42	560.02
32	1024						3982.53	2227.42
40	16	118.65	3221.21			53.68	45.44	49.62
40	32	384.40	*Memory*			119.64	80.95	86.25
40	64	1405.47				451.28	200.89	211.58
40	128	3305.78				1448.10	228.86	247.63
40	256	*Memory*				*Memory*	712.84	555.95
40	512						2393.79	1497.73
40	1024						*Memory*	5711.72

Table 2: Elapsed CPU time in seconds for the well-conditioned problems.

		Serial Codes (SPARCStation 10/51)				**Parallel Splitting (CM–5E)**		
s	N	PATH	SMOOTH	CONOPT	MILES	16 VU	128 VU	256 VU
8	16	2.02	13.96	19.36	6.10	10.91	9.38	9.27
8	32	3.85	23.79	74.53	25.81	14.48	7.04	13.83
8	64	13.96	70.37	344.48	3384.00	23.41	11.52	17.24
8	128	44.12	161.31	*Time*	7541.00	34.10	15.82	23.24
8	256	129.45	950.20		*Time*	107.63	50.42	62.24
8	512	1275.12	*Memory*			901.75	270.01	347.84
8	1024	*Time*				4368.97	1028.12	1039.53
16	16	20.81	138.05	*Failure*		24.44	17.62	22.71
16	32	97.62	392.03	*Failure*		32.45	20.20	31.03
16	64	284.26	774.29	*Failure*		100.06	54.88	64.33
16	128	1243.79	1857.06	*Failure*		111.39	39.74	47.04
16	256	7386.99	*Memory*	*Failure*		322.94	94.44	95.14
16	512	*Time*				1396.56	264.86	275.60
16	1024	*Memory*				*Memory*	1137.17	843.17
16	2048						7225.26	3905.98
24	16	66.85	528.18			67.68	51.75	60.02
24	32	203.28	1477.76			139.57	84.45	98.66
24	64	1645.27	3969.28			494.02	82.51	93.29
24	128	8239.70	*Memory*			966.21	256.65	274.48
24	256	*Time*				1811.61	414.18	380.50
24	512	*Memory*				*Memory*	1493.27	1085.49
24	1024						6682.58	3434.09

Table 3: Elapsed CPU time in seconds for the ill-conditioned problems.

to draw empirical conclusions about our algorithm's abstract speedup potential from the runtime data in Tables 2 and 3. For example, suppose that we wish to compare the 16- and 256-VU data for $N = 256$. One must keep in mind that, even though all the VU's are occupied in both cases, the 16-VU configuration is operating on much longer vectors, and so runs more efficiently. Another way of looking at the situation is that the implementation continues to extract a kind of parallelism from the algorithm even as N increases past the number of processors V, because increasing vector length allows it to increase its use of pipelining internal to each processor. This phenomenon leads to an appearance of sublinear speedup, especially for small s. Larger values of s lead to larger average vector lengths, and thus tend to mitigate the effect somewhat.

4 Conclusion

Sections 1 and 2 have presented a unified approach to set-valued monotone operator splitting theory, and applied it to generate two classes of iterative algorithms for monotone affine variational inequalities. The analysis of these methods does not depend on symmetry, and creates a number of theoretical connections to existing matrix splitting methods. In particular, the iteration (33)-(34) resembles standard matrix splitting interspersed with solutions of certain systems of linear equations, and converges under quite general conditions.

Section 3 has demonstrated that splitting methods are interesting for more than their theoretical connections alone. Splitting allows one to "pull apart" the structure of some problems so they can be attacked in a highly parallel manner. This approach is consonant with the general philosophy of splitting methods, dating back at least to 1950's alternating direction methods for banded systems of linear equations [9, 32]: one expresses the structure of ones problem as a composition of two less complicated structures, each simple enough to be attacked with the technology at hand.

Finally, with careful implementation aimed at circumventing "tail convergence" difficulties, we have demonstrated that these techniques can lead to parallel algorithms that significantly outperform state-of-the-art general solvers on a class of difficult, large-scale affine variational inequality problems. There are many other application areas for which the techniques described in this paper would be appropriate. A particular class of applications for future research is the field of linear-quadratic problems arising in stochastic programming [42, 44].

Acknowledgements: We would like to thank Paul Bay, formerly of Thinking Machines Corporation, for helping us with the CMSSL block tridiagonal linear system routines. Some of the research described here was performed while the first author was employed at Thinking Machines; we would like to thank the firm for the use of their computational facilities.

References

[1] H. Brézis. *Opérateurs Maximaux Monotones et Semi-Groupes de Contractions dans les Espaces de Hilbert.* North-Holland, 1973.

[2] M. Cao and M. C. Ferris. Lineality removal for copositive–plus normal maps. *Communications on Applied Nonlinear Analysis*, 2:1–10, 1995.

[3] C. H. Chen and O. L. Mangasarian. Smoothing methods for convex inequalities and linear complementarity problems. *Mathematical Programming*, 78:51–70, 1995.

[4] G. H.-G. Chen. *Forward–Backward Splitting Techniques: Theory and Applications.* PhD thesis, University of Washington, 1994.

[5] R. W. Cottle and G. B. Dantzig. Complementary pivot theory of mathematical programming. *Linear Algebra and Its Applications*, 1:103–125, 1968.

[6] R. W. Cottle, J. S. Pang, and R. E. Stone. *The Linear Complementarity Problem.* Academic Press, Boston, 1992.

[7] S. P. Dirkse and M. C. Ferris. MCPLIB: A collection of nonlinear mixed complementarity problems. *Optimization Methods and Software*, 5:319–345, 1995.

[8] S. P. Dirkse and M. C. Ferris. The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software*, 5:123–156, 1995.

[9] J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American Mathematical Society*, 82:421–439, 1956.

[10] A. Drud. CONOPT: A GRG code for large sparse dynamic nonlinear optimization problems. *Mathematical Programming*, 31:153–191, 1985.

[11] J. Eckstein. *Splitting Methods for Monotone Operators, with Applications to Parallel Optimization.* PhD thesis, Massachusetts Institute of Techonology, Cambridge, MA, 1989. Report LIDS-TH-1877, Laboratory for Information and Decision Systems, M.I.T.

[12] J. Eckstein and D. P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318, 1992.

[13] M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, C-21:948–960, 1972.

[14] D. Gabay. Applications of the method of multipliers to variational inequalities. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian methods: Applications to the Solution of Boundary Value Problems.* North-Holland, Amsterdam, 1983.

[15] R. W. Hockney and C. R. Jesshope. *Parallel Computers 2.* Adam Hilger, Bristol, 1988.

[16] S. L. Johnsson. Solving tridiagonal systems on ensemble architectures. *SIAM Journal on Scientific and Statistical Computing*, 8:475–489, 1987.

[17] J. Lawrence and J. E. Spingarn. On fixed points of non-expansive piecewise isometric mappings. *Proceedings of the London Mathematical Society*, 55(3):605–624, 1987.

[18] C. E. Lemke. On complementary pivot theory. In G. B. Dantzig and A. F. Veinott, editors, *Mathematics of the Decision Sciences: Part 1*, volume 11 of *Lectures in Applied Mathematics*, pages 95–114. American Mathematical Society, Providence, 1968.

[19] P.-L. Lions. Une Méthode itérative de resolution d'une inequation variationnelle. *Israel Journal of Mathematics*, 31(2):204–208, 1978.

[20] P.-L. Lions and B. Mercier. Splitting methods for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16:964–979, 1979.

[21] P. Mahey, S. Oualibouch, and D. T. Pham. Proximal decomposition on the graph of a maximal monotone operator. *SIAM Journal on Optimization*, 5:454–466, 1995.

[22] P. Mahey and D. T. Pham. Partial regularization of the sum of two maximal monotone operators. *RAIRO Modélisation et Analyse Numérique*, 27:375–392, 1993.

[23] G. J. Minty. Monotone (nonlinear) operators in Hilbert space. *Duke Mathematics Journal*, 29:341–346, 1962.

[24] K. G. Murty. *Linear Complementarity, Linear and Nonlinear Programming.* Helderman-Verlag, Berlin, 1988.

[25] Z. Opial. Weak convergence of the sequence of successive approximations for nonexpansive mappings. *Bulletin of the American Mathematical Society*, 73:591–597, 1967.

[26] J. S. Pang. On the convergence of a basic iterative method for the implicit complementarity problem. *Journal of Optimization Theory and Applications*, 37:149–162, 1982.

[27] J. S. Pang. Necessary and sufficient conditions for the convergence of iterative methods for the linear complementarity problem. *Journal of Optimization Theory and Applications*, 42:1–17, 1984.

[28] J. S. Pang. More results on the convergence of iterative methods for the symmetric linear complementarity problem. *Journal of Optimization Theory and Applications*, 49:107–134, 1986.

[29] J. F. A. D. Pantoja and D. Q. Mayne. Sequential quadratic programming algorithm for discrete optimal control problems with control inequality constraints. *International Journal on Control*, 53:823–836, 1991.

[30] D. Pascali and S. Sburlan. *Nonlinear Mappings of Monotone Type*. Editura Academeie, 1978.

[31] G. B. Passty. Ergodic convergence to a zero of the sum of monotone operators in Hilbert space. *Journal of Mathematical Analysis and Applications*, 72:383–390, 1979.

[32] D. W. Peaceman and H. H. Rachford. The numerical solution of parabolic and elliptic differential equations. *SIAM Journal*, 3:28–41, 1955.

[33] B. T. Polyak. *Introduction to Optimization*. Optimization Software, Inc., Publications Division, New York, 1987.

[34] S. M. Robinson. Normal maps induced by linear transformations. *Mathematics of Operations Research*, 17:691–714, 1992.

[35] R. T. Rockafellar. Characterization of the subdifferentials of convex functions. *Pacific Journal of Mathematics*, 17(3):497–510, 1966.

[36] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.

[37] R. T. Rockafellar. On the maximality of sums of nonlinear monotone operators. *Transactions of the American Mathematical Society*, 149:75–88, 1970.

[38] R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14:877–898, 1976.

[39] R. T. Rockafellar. Monotone operators and augmented Lagrangian methods in nonlinear programming. In O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, editors, *Nonlinear Programming 3*, pages 1–26. Academic Press, London, 1978.

[40] R. T. Rockafellar. Linear-quadratic programming and optimal control. *SIAM Journal on Control and Optimization*, 25:781–814, 1987.

[41] R. T. Rockafellar. Multistage convex programming and discrete-time optimal control. *Control and Cybernetics*, 17(2-3):225–245, 1988.

[42] R. T. Rockafellar and R. J.-B. Wets. A Lagrangian finite generation technique for solving linear-quadratic problems in stochastic programming. *Mathematical Programming Study*, 28:63–93, 1986.

[43] R. T. Rockafellar and R. J.-B. Wets. Generalized linear-quadratic problems of deterministic and stochastic optimal control in discrete time. *SIAM Journal on Control and Optimization*, 28:810–822, 1990.

[44] R. T. Rockafellar and R. J.-B. Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 10:119–147, 1991.

[45] T. F. Rutherford. MILES: A mixed inequality and nonlinear equation solver. Working Paper, Department of Economics, University of Colorado, Boulder, 1993.

[46] Thinking Machines Corporation, Cambridge, MA. *Connection Machine CM–5 Technical Summary*, 1993.

[47] Thinking Machines Corporation, Cambridge, MA. *CM Fortran Language Reference Manual*, 1994.

[48] Thinking Machines Corporation, Cambridge, MA. *CMSSL for CM Fortran*, 1994.

[49] P. Tseng. Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Control and Optimization*, 29:119–138, 1991.

[50] S. E. Wright. Dynfgm: Dynamic finite generation method. Technical report, Department of Mathematics, University of Washington, Seattle, Washington, 1989.

[51] S. J. Wright. Solution of discrete–time optimal control problems on parallel computers. *Parallel Computing*, 16:221–238, 1990.

[52] C. Y. Zhu. On the primal–dual steepest descent algorithm for extended linear–quadratic programming. *SIAM Journal on Optimization*, 5:114–128, 1995.

[53] C. Y. Zhu and R. T. Rockafellar. Primal-dual projected gradient algorithms for extended linear-quadratic programming. *SIAM Journal on Optimization*, 3:751–783, 1993.

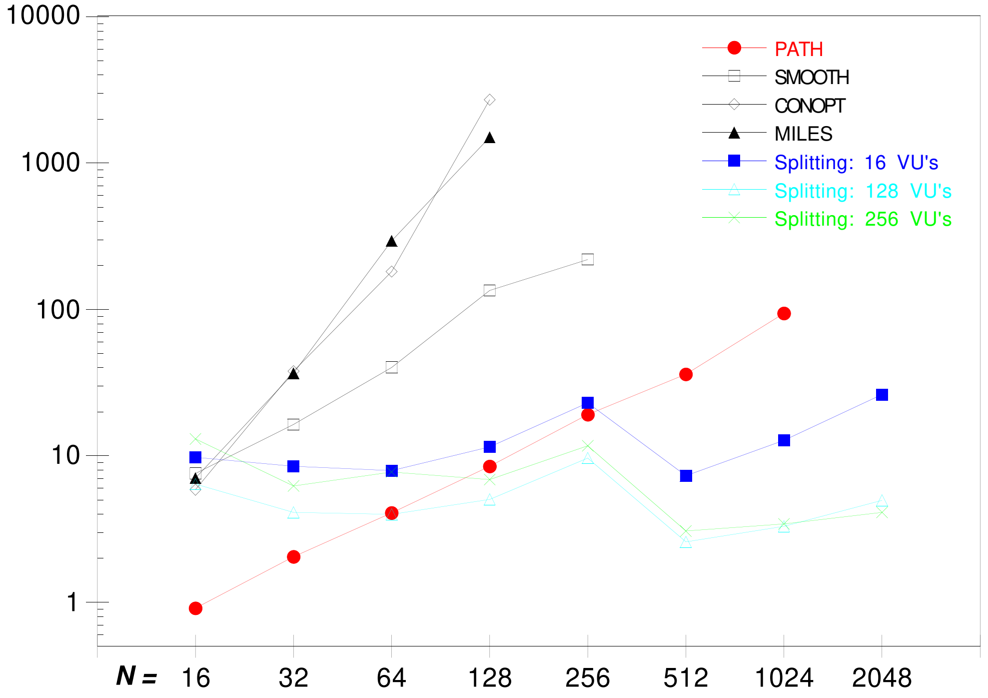

Figure 1: Elapsed CPU time in seconds for well-conditioned problems with $s = 8$.

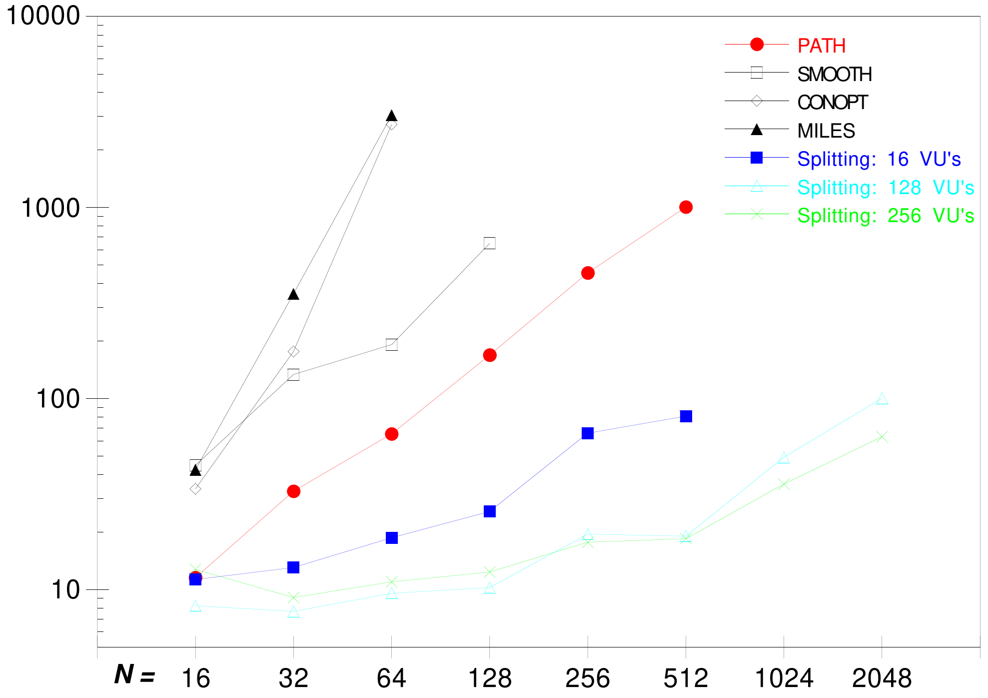

Figure 2: Elapsed CPU time in seconds for well-conditioned problems with $s = 16$.

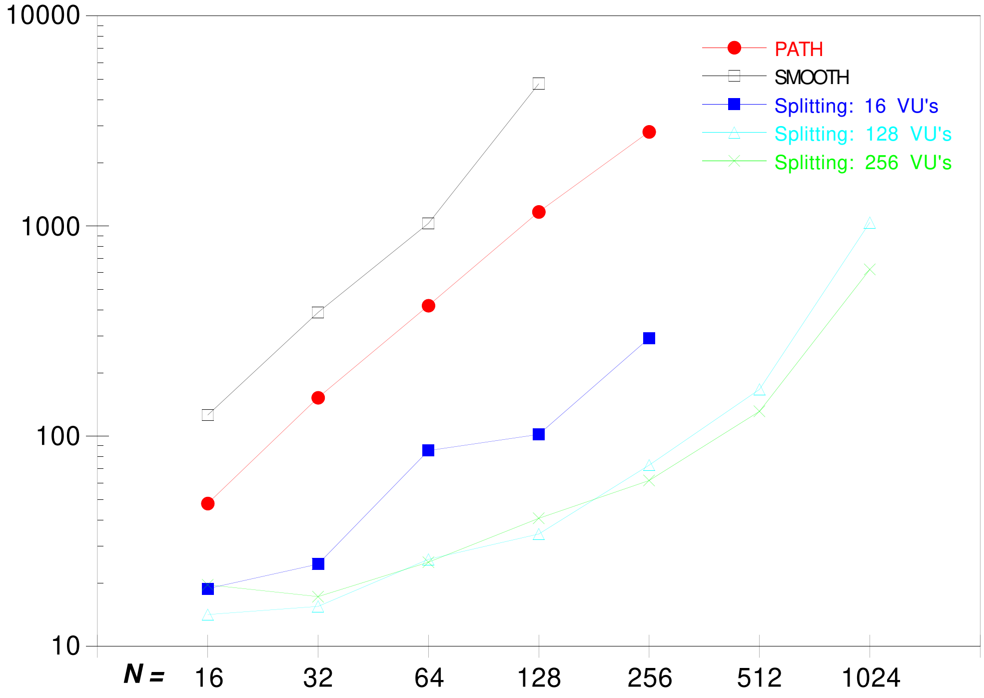

Figure 3: Elapsed CPU time in seconds for well-conditioned problems with $s = 24$.

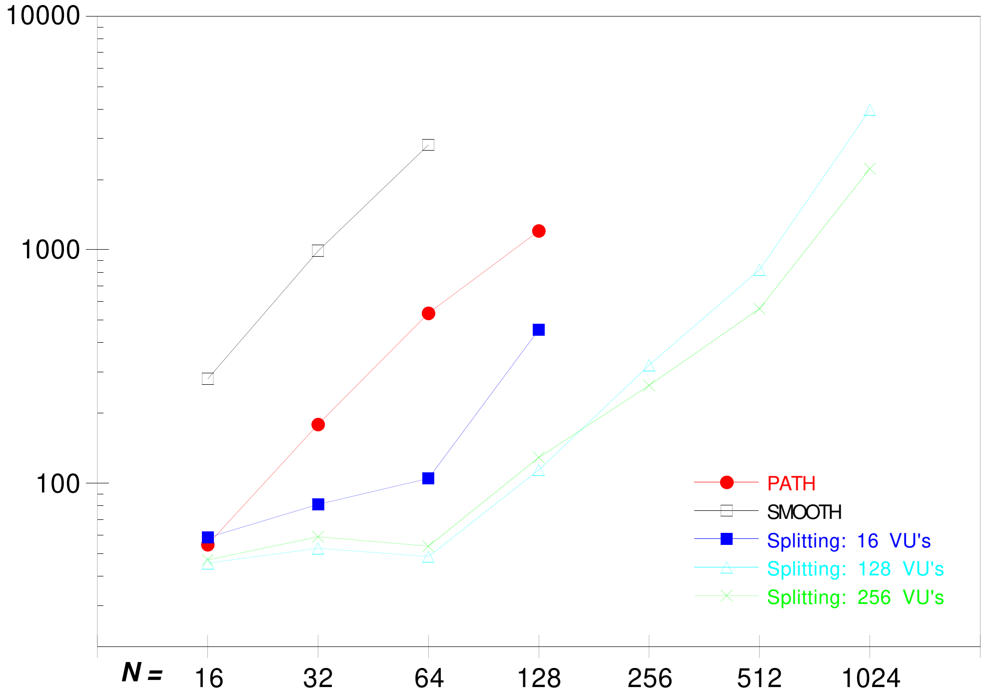

Figure 4: Elapsed CPU time in seconds for well-conditioned problems with $s = 32$.

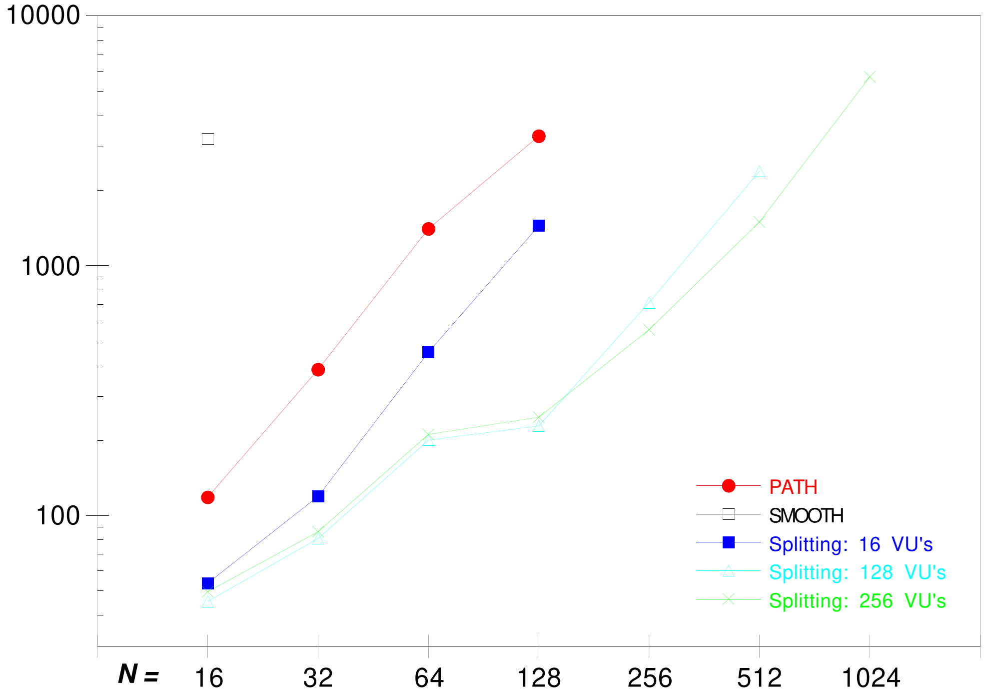

Figure 5: Elapsed CPU time in seconds for well-conditioned problems with $s = 40$.

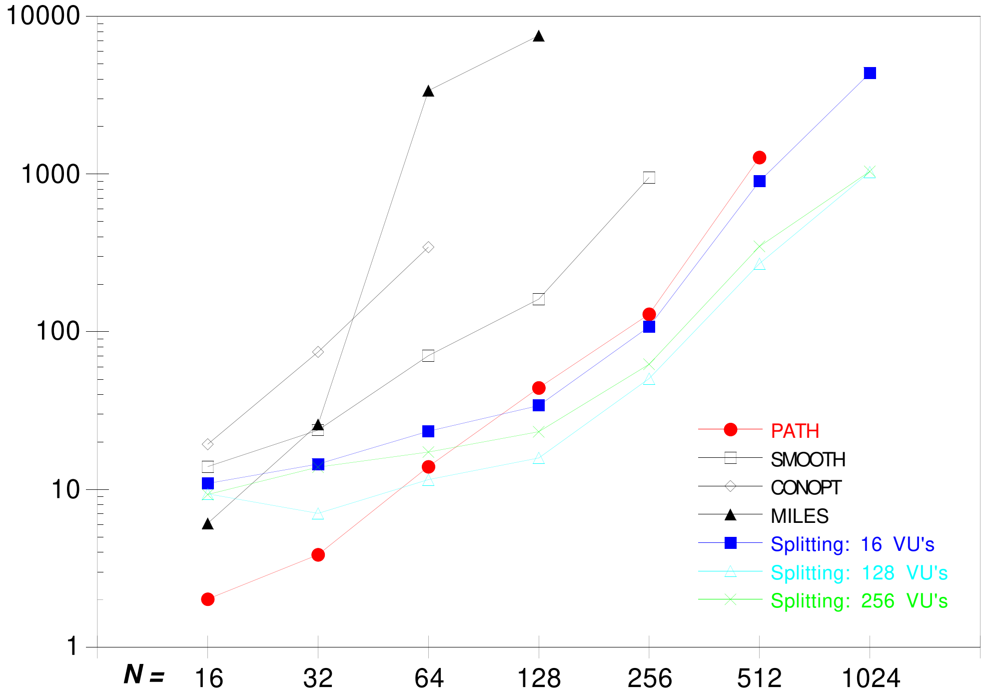

Figure 6: Elapsed CPU time in seconds for ill-conditioned problems with $s = 8$.

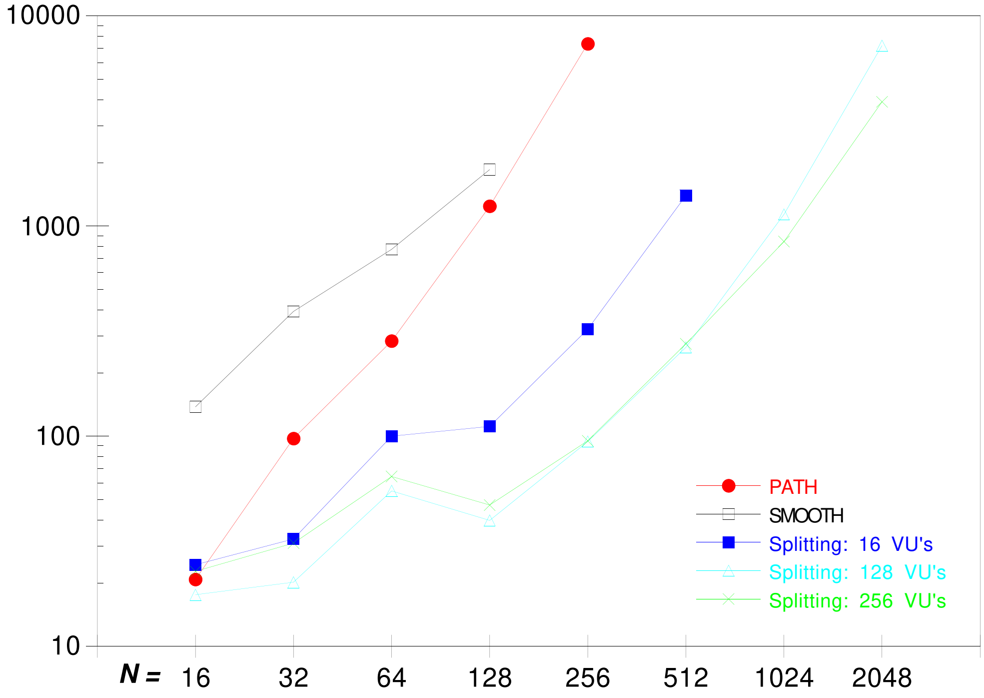

Figure 7: Elapsed CPU time in seconds for ill-conditioned problems with $s = 16$.

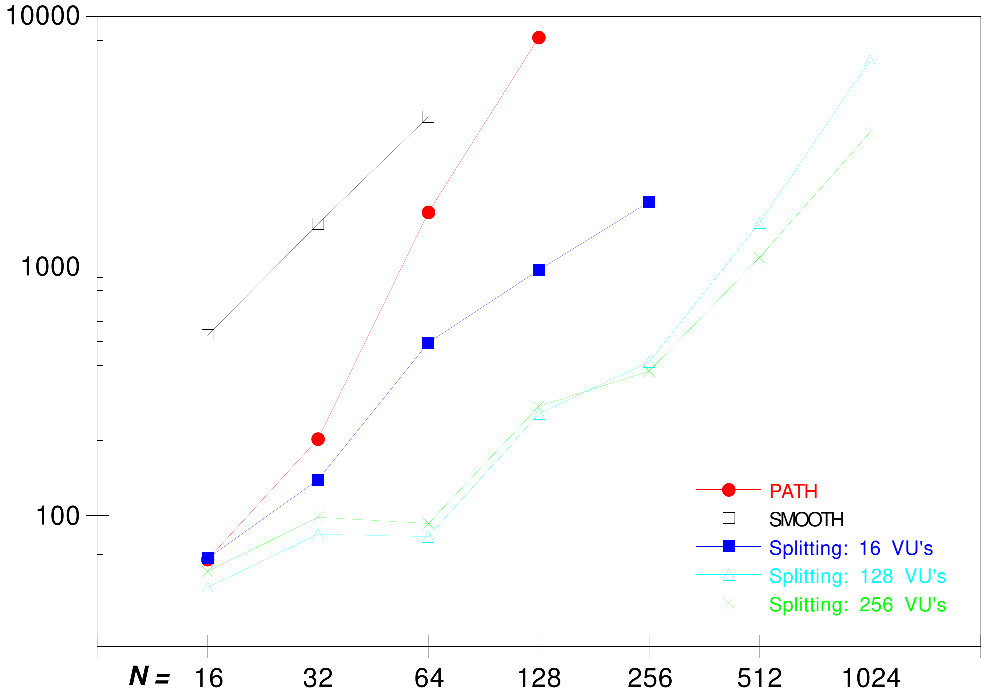

Figure 8: Elapsed CPU time in seconds for ill-conditioned problems with $s = 24$.