June 1989

LIDS-TH-1877

Research Supported By:

Army Research Office
Grant DAAL03-86-K-0171

Splitting Methods for Monotone Operators with Applications to Parallel Optimization

Jonathan Eckstein

June 1989 LIDS-TH-1877

SPLITTING METHODS FOR MONOTONE OPERATORS WITH APPLICATIONS TO PARALLEL OPTIMIZATION

by

Jonathan Eckstein

This report is based on the unaltered thesis of Jonathan Eckstein submitted in partial fulfillment of the requirements of the degree of Doctor of Philosophy at the Massachusetts Institute of Technology in June 1989. This research was carried out at the Laboratory for Information and Decision Systems and Operations Research Center and was supported in part by the Army Research Office under grant number DAAL03-86-K-0171.

Massachusetts Institute of Technology
Laboratory for Information and Decision Systems
Cambridge, MA 02139

Splitting Methods for Monotone Operators with Applications to Parallel Optimization

by

Jonathan Eckstein

A.B., Harvard University (1980)
S.M., Massachusetts Institute of Technology (1986)

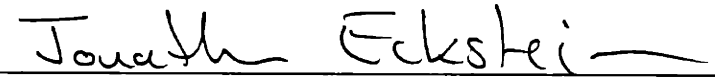
Submitted to the Department of Civil Engineering
in Partial Fulfillment of the Requirements for the Degree of

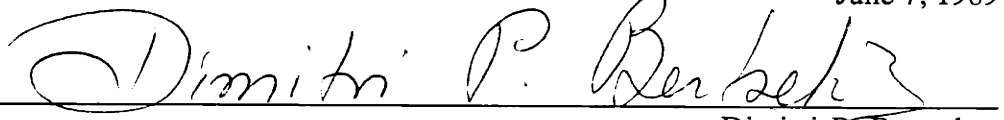
Doctor of Philosophy in Operations Research

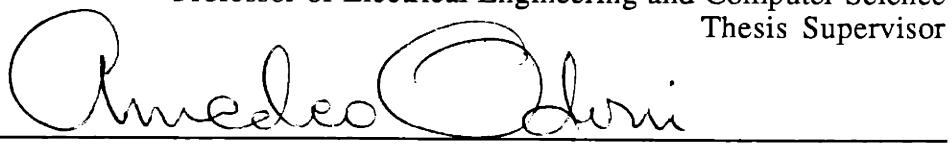
at the

Massachusetts Institute of Technology

September 1989

Signature of Author 
Department of Civil Engineering
June 7, 1989

Certified By 
Dimitri P. Bertsekas
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted By 
Amadeo R. Odoni
Co-Director, Operations Research Center

Splitting Methods for Monotone Operators with Applications to Parallel Optimization

Submitted to the department of Civil Engineering on June 7, 1989,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

The first three chapters of this thesis present a cohesive treatment of monotone set-valued operators in mathematical programming, and the notion of splitting algorithms for finding zeroes of such operators. The idea of splitting unifies a substantial fraction of the recent literature on decomposition methods for convex programming. Chapter 3 presents four kinds of splitting: double-backward, forward-backward, Peaceman-Rachford, and Douglas-Rachford. Of these, Douglas-Rachford is shown to be the most robust. The alternating direction method of multipliers, a method for convex programming, is a special case of Douglas-Rachford splitting.

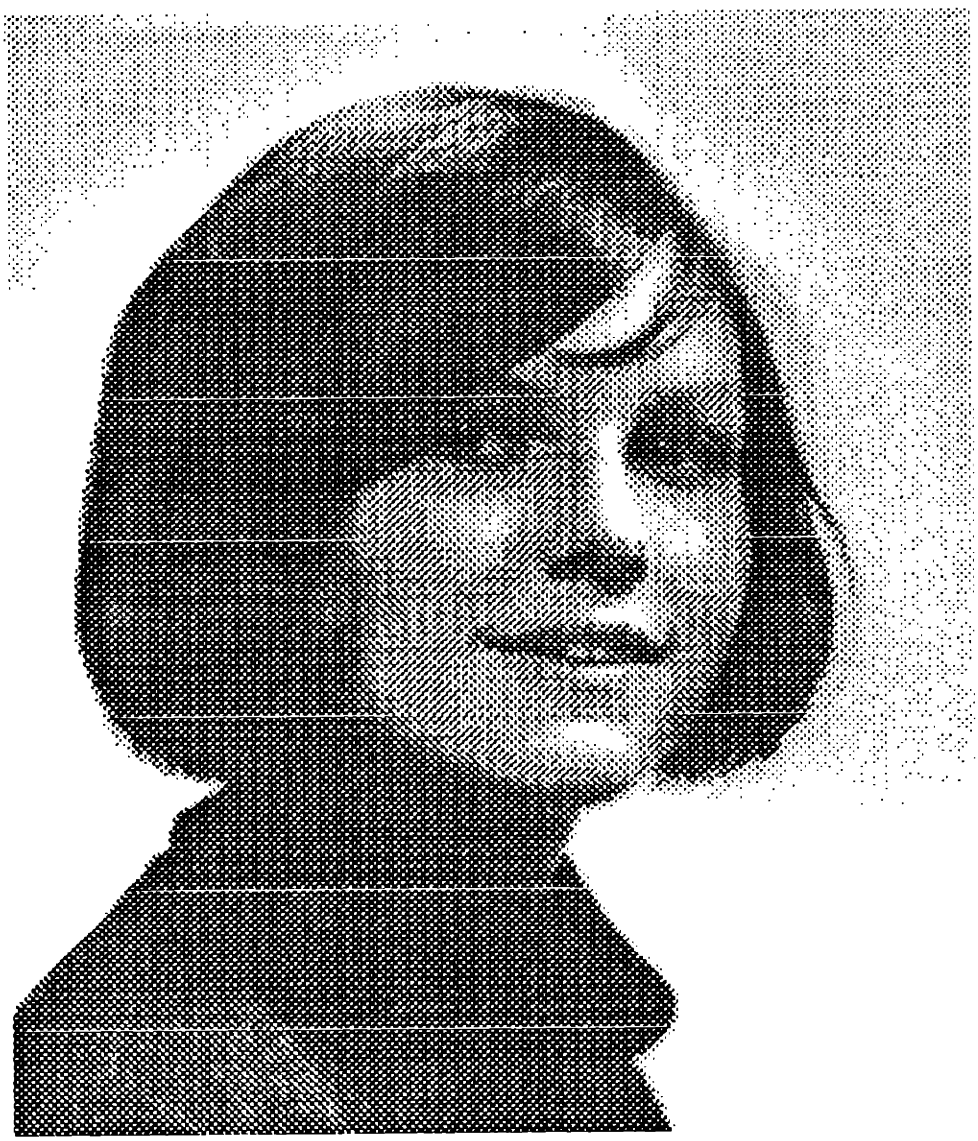
Chapter 4 shows that the Douglas-Rachford splitting scheme is a special case of Rockafellar's proximal point algorithm, and exploits this connection to create a new generalization of the alternating direction method of multipliers. Chapter 5 uses the alternating direction method of multipliers to derive three new convex programming decomposition methods. The first is an algorithm for problems with multiple set constraints, the second is the *epigraphic projection method* for problems with block-separable structure, and the third is the *alternating step method* for monotropic programming.

When specialized to linear programming, the alternating step method yields a novel, highly parallel algorithm which is unusual in that it does not maintain primal feasibility, dual feasibility, or complementary slackness. Chapter 6 studies this method, and, drawing on the theory of the proximal point algorithm, shows that it converges at a linear rate. Chapter 7's detailed computational tests, performed on several kinds of parallel computers, reveal that the alternating step method shows some promise of being useful for assignment problems.

Thesis Supervisor: Dimitri P. Bertsekas
Title: Professor of Electrical Engineering and Computer Science

Dedication

To my late cousin Margaret, a person of great perseverance and courage.



Margaret Fjeld 1949-1989

Acknowledgements

I would first like to thank my thesis advisor, Dimitri Bertsekas, without whom anything even resembling this work would not have been possible. I also thank the other members of my thesis committee, Tom Magnanti and John Tsitsiklis.

This Research was supported in part by the Army Research Office, under grant number DAAL03-86-K0171, through the Harvard/MIT/Brown Center for Intelligent Control Systems (CICS).

The Mathematics and Computer Science Division and the Advanced Computing Research Facility (ACRF) of the Argonne National Laboratory made very generous donations of computer time and support; particular thanks should go to David Levine and Brian Smith. I am also grateful to Sandy Pentland of the MIT Media Lab for making time available on a Connection Machine 2.

A number of people outside my official thesis committee also made helpful suggestions or asked useful questions during the course of my research. These people include Paul Tseng, Rob Freund, David Castañon, Stavros Zenios, and Terry Rockafellar. I thank them for their interest in my work. I would also like to thank Paul and Rob for being "unofficial" members of my thesis committee.

On the personal side, I would like to thank my fiancée Bonnie, my mother Joan, and my father Harry for their tremendous support and forbearance. I should also mention the support given by my friends Tom Levenson and Brian Pentland. Brian, and also Scott Mitchell, Marian Hubler, and Andy Trice made for many pleasurable musical breaks from the fray (both with and without a certain basement band that shall remain nameless).

Finally, I would like to thank Marcia Chapman — the great slayer of bureaucratic demons — and the other students and staff who make the OR Center and LIDS pleasant places to work.

Table of Contents

Summary of Notation

Sets, scalars, general operators, and members of general Hilbert spaces are denoted by *italics*. Matrices and finite-dimensional vectors are represented by **boldface**, and their components by subscripted italics. A superscript "k", "$k+1$", or "$k-1$" generally denotes an iteration number. Boldface letters with subscripts generally denote subvectors or submatrices of previously defined vectors or matrices.

$X^{\perp}$ denotes the orthogonal complement of the linear subspace X.

$\langle x, y \rangle$ denotes the inner product of x and y.

$\|x\|$ or $\|x\|_2$ denotes the Euclidean norm $\langle x, x \rangle^{1/2}$ of x. In finite-dimensional spaces, $\|\mathbf{x}\|_\infty$ denotes the l_∞-norm $\max_j \{|x_j|\}$, and $\|\mathbf{x}\|_1$ the l_1-norm $\sum_j |x_j|$.

If y is a point and X is a set, then $\mathrm{dist}(y, X)$ denotes $\inf \{\|y - x\| \mid x \in X\}$, and $\mathrm{dist}^2(y, X)$ denotes $(\mathrm{dist}(y, X))^2$.

The notation $T: X \rightrightarrows Y$ means that T is a multivalued map from X to Y, as defined in Section 3.1.

If h is a function, im h denotes its range (or "image"). If h is differentiable at $\mathbf{x}$, $\nabla h(\mathbf{x})$ denotes its gradient at $\mathbf{x}$, represented as a column vector.

Given a matrix $\mathbf{M}$, im $\mathbf{M}$ denotes its column space, and $\mathcal{N}(\mathbf{M})$ its null space (kernel).

Given a set C, int(C) denotes its interior, ri(C) denotes its *relative* interior, cl(C) denotes its closure, and bd(C) denotes its boundary.

The $\circ$ symbol denotes functional composition.

The symbol $^{\mathsf{T}}$ denotes transposition. "Prime" marks are *not* used to denote the transpose.

The symbol $\stackrel{\Delta}{=}$ should be read as "defined to be equal to".

A backslash denotes the set difference operation, that is, $A \backslash B = \{x \in A \mid x \notin B\}$.

All further notation is either standard, or defined in the text.

Chapter 1

Introduction

One goal of current mathematical programming research is to identify algorithms that will make efficient use of massively parallel computer architectures. One basic approach is to apply techniques of *decomposition* — dividing a problem into many smaller ones that can be solved in parallel. This thesis will concentrate on techniques leading to *fine-grain* or *massive* parallelism. Conventional mathematical programming decomposition methods, such as Dantzig-Wolfe decomposition (Dantzig 1963) and Benders' decomposition (Benders 1962), typically require specialized block-angular or dual block-angular constraint structures, and generally lead to a relatively coarse-grained partition of the original problem. The methods we will discuss here will make weaker assumptions, or in the case of linear programming, virtually no assumptions as to problem structure. One should note, however, that the techniques presented in this thesis could be used to derive coarse-grained decomposition methods, as well as fine-grained ones.

The decomposition principle which we will use, although conceptually simple, has not been broadly recognized in the American mathematical programming community. The principle is that of "splitting" a monotone operator by expressing it as the sum of two simpler operators, and then exploiting the resulting structure. After the literature review of Chapter 2, Chapter 3 will give a unified development of this decomposition technique, and show how a variety of existing optimization methods — and some new ones — fit into the operator splitting framework. Chapter 4 will focus attention on a particularly

robust splitting scheme, the *Douglas-Rachford* method of Lions and Mercier (1979), and show that it is an application of the proximal point algorithm (Rockafellar, 1976a) to a special monotone operator we call the *splitting operator*. This observation, although it follows without great difficulty from Lions and Mercier's original paper, appears to be new, and permits a unified treatment of a number of recently proposed decomposition methods. These methods include the *partial inverse* work of Spingarn (1983, 1985b), some recent work of Gol'shtein (1985, 1986, 1987), and the *alternating direction method of multipliers* due to a number of French authors (Glowinski and Marroco 1975, Gabay and Mercier 1976, Fortin and Glowinski 1983, Gabay 1983, Glowinski and Le Tallec 1987, Bertsekas and Tsitsiklis 1989). The relationship between Douglas-Rachford splitting and the proximal point algorithm permits the derivation of new algorithms, in particular the *generalized* alternating direction method of multipliers.

Chapters 5 and 6 focus on the alternating direction method of multipliers, which in the past has not been explicitly applied to the central problems of mathematical programming. We attempt to rectify this oversight, giving novel applications to convex, and, in particular, linear programming. One linear programming algorithm, which we will call the *alternating step method*, consists of a very simple iteration that lends itself naturally to parallel computation. It has the unusual feature that it does not maintain *any* of the three customary invariants of primal feasibility, dual feasibility, or complementary slackness. Chapter 6 explores the alternating step method for linear programming in detail. Using the concept of the splitting operator, we will apply ideas from the established theory of the proximal point algorithm to show that the convergence rate of the alternating step method is linear when the method is applied to linear programs in standard primal form.

The main practical question about the alternating step method is under what circumstances its convergence rate is fast enough to make it competitive. Chapter 7 presents computational experiments that begin to address this issue. For most NETGEN-generated network problems (Klingman *et. al.* 1974), the present form of method has a tendency to converge disappointingly slowly, exhibiting a curious "spiralling" behavior. However, we will show that for *assignment* problems, its convergence is often quite satisfactory. We conclude with some experiments on actual parallel computers. The main difficulty in implementing the algorithm is that while it permits a great deal of concurrency, it also requires a considerable amount of communication between processors. On the current generation of parallel processors, which communicate far less rapidly than they compute, communication thus appears to become a bottleneck.

Chapter 8 presents conclusions and some possible topics for further research.

Chapter 2

Literature Review

2.1. Monotone Operators and the Proximal Point Algorithm

The topic of monotone operators has a significant history, and its roots seem to be primarily in functional analysis, rather than mathematical programming. A complete literature review would be prohibitively long, but important early contributions and surveys may be found in Browder (1964, 1965, 1968), Minty (1961, 1962, 1964), Kachurovskii (1968), and Rockafellar (1966, 1969). The original definition of a monotone operator appears to have been formulated by Kachurovskii (1960). Two books on the subject are by Brézis (1973) and Pascali and Sburlan (1978), which contain numerous applications to functional analysis, engineering problems, and mathematical physics. Doležal's book (1979), which has an abstract control theory orientation, is another useful reference. Chapter 3 of Joshi and Bose (1985) also gives a concise development of much of the essential theory, as does the first section of Brézis (1973). Supporting material may be found in Debrunner and Flor (1964), Fan (1952), and Glicksberg (1952). The Fan and Glicksberg results depend on Kakutani's (1941) fixed point theorem.

Further, somewhat less pertinent material on monotone operators may be found in Deimling (1985), where set-valued operators are called *multis*, Cesari (1983), Zeidler (1985), and Vaĭnberg (1973). Kenderov (1974) proved that monotone operators are almost everywhere single-valued.

The proof that the resolvent $J_{\lambda T} = (I + \lambda T)^{-1}$ of a maximal monotone operator T is single-valued everywhere defined may be traced back to Minty (1962). The proof requires Zorn's Lemma (see, for example, Dugundji, 1966), which in turn is equivalent to the axiom of choice. Thus, many of the results of this thesis will depend indirectly on the axiom of choice.

We will mainly consider special cases in which the resolvent operator is relatively easy to compute. Bruck (1973) proposed a general iterative method for evaluating the resolvent. The resolvents of monotone operators have a *firmly nonexpansive* (sometimes called *pseudocontractive*) property that will be stressed in Chapters 3 and 4. The symmetry between monotone and firmly nonexpansive operators is not prominent in the literature, but Browder and Petryshyn (1967) have treated, in passing, the single-valued case. For further material on firmly nonexpansive operators, see Martinet (1972).

Rockafellar's classic book (1970a) contains indispensable material on monotone operators, as well as on convex analysis concepts used repeatedly in this thesis. Another source on convex analysis is Ekeland and Temam (1976). Rockafellar's more recent book (1984), by essentially considering only the single-dimensional case, presents a simplified treatment of monotone subgradient operators that still conveys many of the essential ideas.

Many key results regarding the role of monotone operators in mathematical programming are due to Rockafellar (1966, 1969, 1970a-d, 1976a-b, 1977, 1984). He established conditions for the maximal monotonicity of subdifferential mappings (1970b), and the maximality of sums of operators (1970c). He codified (1976a) the general *proximal point algorithm* for finding a zero of a maximal monotone operator, building on specialized

versions devised earlier by Martinet (1970, 1972). The term "proximal point" was originally coined (in French) by Moreau (1962, 1965). The proximal point iteration will be a key algorithmic building block in this thesis.

Brézis and Lions (1978) made some refinements to the convergence proof of the proximal point algorithm, while Gol'shtein (1975, see also Gol'shtein and Tret'yakov 1979) has shown the convergence of under- and over-relaxation variants of the method.

The convergence *rate* analysis of the proximal point algorithm was sharpened by Luque (1984a); we will use similar ideas to analyze the convergence rate of the alternating step method. Luque also developed "nonlinear" generalizations of the proximal point algorithm in (1984b, 1986a-c), similar to some work of Gol'shtein (1975) and of Gol'shtein and Tret'yakov (1979). Related results for variants of the method of multipliers may be found in Kort and Bertsekas (1972, 1973, 1976). Unfortunately, there does not appear to be any tractable application of the nonlinear proximal point algorithm to operator splitting.

In Chapter 3, we will treat the gradient, subgradient, and proximal point algorithms as approximate path-following algorithms. This idea, at least for gradient methods, appears to date back to Arrow *et. al.* (1958). Polyak (1970) has also explored such ideas, as have Botsaris and Jacobson (Botsaris and Jacobson 1976, Botsaris 1978a, 1978b). Bruck (1975b) has shown the existence of the steepest-descent paths under much weaker assumptions than in Botsaris and Jacobson's work. Interest in differential path-following algorithms for optimization has received renewed interest with the appearance of Karmarkar's algorithm and its relatives; the literature of this family of methods is too extensive to discuss here. The class of path-following methods in Garcia and Zangwill's book (1981) is of a different sort.

2.2. *Decomposition and the Method of Multipliers*

The preeminent use of the proximal point algorithm in mathematical programming is the *method of multipliers*, due independently to Hestenes (1969) and Powell (1969). Rockafellar (1976b) showed that in the case of convex problems, the method of multipliers may be obtained by applying the proximal point algorithm to the subgradient of the dual functional. In the same paper, he introduced a primal application of the proximal point algorithm to convex programming, the *proximal minimization algorithm*, and a combined primal-dual application, the *proximal method of multipliers*. The method of multipliers remains the best-known of these three methods, and has an extensive literature. Bertsekas (1982) gives an comprehensive treatment of this method. Bertsekas (1975) also proved that the method of multipliers converges in a finite number of iterations for linear programming.

The method of multipliers is a proven optimization method, but its use in large-scale problems has been hampered because it employs an *augmented Lagrangian*, that is, a Lagrangian function to which a (typically) quadratic penalty term has been added. The difficulty is that this additional term is usually not separable, so that in cases where the task of minimizing the usual Lagrangian can be decomposed into a number of independent problems, that of minimizing the augmented Lagrangian cannot. In situations requiring decomposition, the method of multipliers therefore becomes difficult to apply. This situation is unfortunate, as the method of multipliers has desirable convergence properties that methods using the conventional Lagrangian lack.

There are a number of approaches to overcoming the inseparability of the augmented Lagrangian. Stephanopoulos and Westerberg (1975) suggested replacing the penalty

term with a separable, dynamically updated linear approximation. Cohen and Zhu (1984) have explored similar ideas. Watanabe *et. al.* (1978) recommended a different procedure, in which the penalty term is rewritten as a minimum of a sum of separable functions that involve additional variables. Ha (1980) studied a simpler approach for convex programs with a primal block-angular structure. In Ha's approach, one applies the *proximal minimization* algorithm to the primal problem, yielding a (potentially infinite) sequence of problems involving *separable* quadratic penalty terms, and then applies conventional Lagrangian decomposition to each problem in the sequence. This approach inherits some of the advantages of the method of multipliers, but like the methods of Stephanopoulos, Westerberg, Watanabe *et. al.*, at the cost of introducing an additional level of subproblems to be solved. Ha's approach resembles that of an earlier paper by Bertsekas (1979a), who used the proximal minimization algorithm as part of a convexification-decomposition procedure for nonconvex optimization.

Ruszczynski (1988a) has recently introduced a Dantzig-Wolfe-like method that resembles the method of multipliers, yet permits decomposition without introducing an additional optimization loop. He has also developed a method dual to this algorithm (Ruszczynski 1986) and has worked on applications (1988b). Auslender (1985) has also published work in which Dantzig-Wolfe column generation ideas are combined with augmented Lagrangians.

A more radical approach to overcoming the inseparability of the augmented Lagrangian is embodied in the *alternating direction method of multipliers*, in which one does not truly minimize the augmented Lagrangian. A full explanation and derivation of this method will be given in Chapter 3. Most work on the alternating direction method of multipliers has been done by French researchers. It first appeared in Glowinski and Marroco (1975),

where it was proposed to solve a finite element problem. Another early application was in Gabay and Mercier (1976). More thorough theoretical treatments then appeared in Fortin and Glowinski (1983), Gabay (1983), and Glowinski and Le Tallec (1987). Bertsekas and Tsitsiklis (1989) also present the method. In general, the French researchers who developed the alternating direction method of multipliers do not seem to have applied it to the "core" problems of mathematical programming, but rather to numerical problems arising in physics and engineering. One goal of this thesis is to try to fill this gap in the development of the method.

2.3. Splitting Methods for Monotone Operators

The key result in Gabay (1983) is that the alternating direction method of multipliers is an application of the "Douglas-Rachford" *splitting* method for finding a zero of the sum of two maximal monotone operators, as proposed by Lions and Mercier (1979). This insight is the principal inspiration for the first part of this thesis. Lions and Mercier's 1979 paper is also critical to the development of Chapter 3.

In addition to Douglas-Rachford-based splitting for monotone operators, Lions and Mercier also analyzed *Peaceman*-Rachford schemes; a precursor to their analysis appears in Kellogg (1969). More recently, Lin and Cryer (1985) have applied this kind of approach to certain linear complementarity problems.

In addition to work with Mercier, Lions (1978) also explored a different operator splitting scheme, which we call the *double-backward* method. Similar, more general results can be found in Passty (1979). This type of procedure is reminiscent of the classical linear inequality algorithms of Agmon (1954), and of Motzkin and Schoenberg (1954).

Another operator-splitting method is the *forward-backward* scheme. This approach generalizes the gradient projection method for convex optimization, originally proposed by A. A. Goldstein (1964), and independently by Levitin and Polyak (1966). The lesser-known *subgradient* projection method (*e.g.* Polyak 1987) is also an example of forward-backward splitting. Projection schemes have also been quite prominent in the study of variational inequality problems (*e.g.* Sibony 1970, Dafermos 1980, 1983, Bertsekas and Gafni 1982). Further special cases of forward-backward splitting have been examined by Bruck (1975a, 1977). Passty (1979) and Gabay (1983) have also considered this class of methods more generally. Tseng (1988) has also studied the general form of the algorithm, providing more detailed proofs than Gabay, and giving applications to convex programming and variational inequalities. In particular, he has shown that a parallel decomposition algorithm recently proposed by Han and Lou (1988) is an application of forward-backward splitting.

Before Lions and Mercier's 1979 work, the general principle of operator splitting was well-known in the fields of numerical linear algebra and discretized differential equations. In these fields, however, the operators are virtually always linear and single-valued, and the splittings usually reflects the action of an operator along two or three different spatial coordinates. The literature associated with these aspects of splitting is too voluminous to discuss fully here; some seminal papers are Peaceman and Rachford (1955), Douglas and Rachford (1956), and Douglas and Gunn (1964). Marchuk's book (1975) contains several sections on "splitting-up" methods and a large bibliography of operator splitting results, mostly relating to partial differential equations. Ferziger (1981) is a more application-oriented reference.

In Chapter 3, we outline the relationship between a number of optimization algorithms and a variety of alternating direction, splitting, or "splitting-up" methods for solving differential equations. This furthers a line of analysis found not only in Gabay (1983), but also in Glowinski and Le Tallec (1987). The latter paper uses a finite-difference representation of the splitting schemes, as opposed to the (possibly more illuminating) operator-theoretic representation of Lions and Mercier (1979).

2.4. Partial Inverses

Spingarn (1983) has recently introduced the concept of the *partial inverse* of a monotone operator, and given applications to finding the zero of the sum of an arbitrary number of monotone operators (1983), solving systems of linear inequalities (1985a), minimizing a convex function over a linear subspace (1985b), and decomposing block-separable convex programs (1985b). For convex problems, the recent "progressive hedging" stochastic programming scheme proposed by Rockafellar and Wets (1987) is a special case of the partial inverse method for minimizing over a subspace. In Chapter 4, expanding on Eckstein (1988), we will demonstrate that the method of partial inverses is a special case of the "Douglas-Rachford" approach to operator splitting of Lions and Mercier. Therefore, all methods based on partial inverses, including progressive hedging, are manifestations of the same decomposition principle that yields the alternating direction method of multipliers. Chapter 5 will introduce an *epigraphic projection* method for block-separable convex programs, again based on the same principle, that is similar to Spingarn's block-separable method, but should have more possibilities for parallelism.

Lefebvre and Michelot (1988) have given some conditions under which the method of partial inverses converges in a finite number of iterations, and refer to some specific

examples in which it does not (Durier and Michelot 1986). In Chapters 6 and 7, we will give examples of infinite convergence which are much simpler.

Recently, Gol'shtein (1985, 1986, 1987) has proposed a variety of "block" decomposition methods for convex programming, based on a somewhat complicated monotone operator algorithm (Gol'shtein 1987). We will show that this algorithm is also a special case of the Douglas-Rachford splitting. The basic idea used in demonstrating the equivalence is very similar to that of Spingarn's method (1983) for finding a zero of a finite sum of monotone operators.

2.5. Miscellaneous Related Work

In contrast to the rich literature related to the general technique of decomposition via operator splitting, there is essentially no preexisting work relating to the alternating step method for linear programming. The method has already appeared in print (Bertsekas and Tsitsiklis 1989) in material written with the aid of the author.

When applied to network problems, the alternating step method bears an interesting resemblance to the ϵ-relaxation method for minimum cost flow (Bertsekas 1986, Bertsekas and Eckstein 1987, 1988), which is essentially the same as the "inner loop" of the Goldberg-Tarjan minimum cost flow algorithms (Goldberg 1987, Goldberg and Tarjan 1987). These algorithms, their maximum flow counterparts, and their extensive variations have created another body of research that is too large to mention here. In them, the dual variables (node prices) can only be adjusted in one direction (by convention, upward), flows and prices can only be adjusted at nodes with (by convention) positive flow imbalance, and flows must be adjusted in accordance with the principle of ϵ-complemen-

tary slackness (Bertsekas 1979, 1985, 1986, Tardos 1985). By contrast, in the alternating step method as applied to networks, the rules for modifying the flow variables are quite different, and a node's price moves up and down in proportion to the amount of flow imbalance there. Professor Bertsekas first suggested using some form of alternating direction method when we were trying to design a variant of ε-relaxation that would adjust prices both upwards and downwards. However, the alternating step method turns out to be applicable to any linear program, whereas the ε-relaxation class of methods have so far resisted extension even to problems closely related to minimum cost flow, such as minimum cost flow with positive gains.

It is also interesting to compare the alternating step method, as applied to separable, strictly convex-cost optimization on networks, to the Jacobi version of the relaxation algorithm of Bertsekas and El Baz (1987), as well as to the related algorithm of Tseng, Bertsekas, and Tsitsiklis (1988). Again, the relaxation algorithm requires a network constraint structure, while the alternating step method does not.

In examining the convergence rate of the alternating step method, we will draw on the existing stability literature for linear programming. Stability theory concerns the question of how the optimal primal and dual solution sets of a linear program — as opposed to just the optimal objective value — change in response to perturbations in the cost and right-hand-side vectors. Such questions have been addressed by Hoffman (1952), Williams (1963), Robinson (1977), Mangasarian (1981), Cook *et. al.* (1986) — with a similar development in Schrijver (1986) — and Mangasarian and Shiau (1987). The convergence rate analysis presented here constitutes an apparently novel use of such results.

Previous convergence rate results for a method based on Douglas-Rachford splitting include Spingarn's (1985b) analysis of his block-separable convex programming algorithm, and Rockafellar and Wets' (1987) analysis of progressive hedging. Spingarn's analysis assumes that both the primal and dual solutions of the convex program are unique. The linear programming convergence rate proofs given in Chapter 6 make no such assumptions.

Chapter 7 discusses implementation of the alternating step method on a variety of real parallel computers, with a concentration on linear network problems. Related work has been done by Zenios (1988b). The representation scheme for sparse networks on the *Connection Machine 2* multiprocessor was devised by Zenios and Lasken (1988). An extensive bibliography of parallel optimization papers may be found in Zenios (1989). Zenios (1988a) has also described the workings of a variety of multiprocessor systems from a numerical optimization point of view.

Chapter 3

Monotone Operators and the Splitting Principle

This chapter lays the theoretical foundations of this thesis, and has two purposes. First, it gives an overview of relevant portions of the theory of monotone operators, the proximal point algorithm, and convex analysis. Second, it motivates the notion of splitting methods, and demonstrate how splitting schemes can yield a variety of decomposition algorithms for convex optimization. Some of the algorithms obtained are old, and some are new. Only a minority of the material here is original; the main contribution of this chapter is in the organization and presentation.

3.1 Monotone Operators, Graphs, and Subgradients

Most texts on set-valued mappings make a distinction between mappings and their graphs. We will make no such distinction:

Definition 3.1. Given any two sets X and Y we will say that a (possibly multivalued) *mapping* $X \rightrightarrows Y$ is simply any subset of $X \times Y$. If A is such a mapping and $x \in X$, we write Ax or $A(x)$ to denote the set $Ax = \{y \in Y \mid (x, y) \in A\}$.

If A is single-valued, that is, the cardinality of Ax is at most 1 for all $x \in X$, we will by slight abuse of notation allow Ax and $A(x)$ to stand for the unique $y \in Y$ such that $(x, y) \in A$, rather than the singleton set $\{y\}$. The intended meaning should be clear from context.

Other authors may define a mapping $A: X \rightrightarrows Y$ to be a function $A: X \to 2^Y$, and let the *graph* of A be $G(A) = \{(x, y) \mid y \in A(x)\}$. Our notation is essentially equivalent, but permits some streamlining of proofs.

In the literature, multivalued mappings are also called *multis* (Deimling 1985) and *multifunctions*.

Definition 3.2. The *domain* of a mapping A is its "projection" onto the first coordinate,

$$\operatorname{dom} A = \{x \in X \mid \exists\, y \in Y : (x, y) \in A\} = \{x \in X \mid Ax \neq \varnothing\}.$$

We say that A has *full domain* if $\operatorname{dom} A = X$. The *range* or *image* of A is similarly defined as its projection onto the second coordinate,

$$\operatorname{im} A = \{\, y \in Y \mid \exists\, x \in X : (x, y) \in A\,\}\,.$$

Definition 3.3. The *inverse* $A^{-1}: Y \rightrightarrows X$ of a mapping $A: X \rightrightarrows Y$ is $\{(y, x) \mid (x, y) \in A\}$. Given two mappings $T: X \rightrightarrows Y$ and $R: Y \rightrightarrows Z$, their *composition* RT or $R \circ T$ is the $X \rightrightarrows Z$ mapping given by

$$RT = \{\, (x, z) \mid \exists\, y \in Y : \ (x, y) \in T,\ (y, z) \in R\ \},$$

hence

$$(RT)x = \bigcup_{y \in Tx} Ry\,.$$

The of inverse operator given in Definition 3.3 is sometimes called a *pseudoinverse* in the literature (*e.g.* Doležal 1979), because there is no requirement that A correspond to an injective function, or, equivalently, that A^{-1} be single-valued. However, the term *pseudoinverse* has competing definitions, and we will avoid it.

Definition 3.4. Suppose that X is any set and Y is a vector space over the real numbers. then

$$cA = \{(x, cy) \mid (x, y) \in A\} \qquad \forall\, c \in \mathbb{R}, A: X \rightrightarrows Y$$

$$A+B = \{(x, y+z) \mid (x, y) \in A,\ (x, z) \in B\}, \qquad \forall\, A,B: X \rightrightarrows Y\,.$$

Note that the set of mappings $A: X \rightrightarrows Y$ does not itself comprise a true vector space under these definitions. We now place additional structure on X and Y.

Definition 3.5. Let $\mathcal{H}$ be a Hilbert space over the reals with inner product $\langle \cdot\,, \cdot \rangle$ and the corresponding norm topology. An *operator* on $\mathcal{H}$ is any mapping $\mathcal{H} \rightrightarrows \mathcal{H}$. We will always use I to denote the *identity* operator $\{(x, x) \mid x \in \mathcal{H}\}$.

Definition 3.6. Let $\mathcal{H}$ be a real Hilbert space with inner product $\langle \cdot\,, \cdot \rangle$. An operator $T: \mathcal{H} \rightrightarrows \mathcal{H}$ is *monotone* if

$$\langle x' - x, y' - y \rangle \geq 0 \qquad \forall\, (x, y), (x', y') \in T\ .$$

A *maximal* monotone operator is a monotone operator that is not strictly contained in any other monotone operator.

Monotone operators of this type are also called *monotone sets* (Debrunner and Flor 1964) or *totally-M-related sets* (Minty 1962).

We give some basic examples of monotonicity-preserving transformations of operators.

Proposition 3.1. Let $T: \mathcal{H} \rightrightarrows \mathcal{H}$ be an operator on a real Hilbert space. Then

(i) (*Scaling*) Let r be any positive real number. Then T is (maximal) monotone if and only if rT is (maximal) monotone.

(ii) (*Inversion*) T is (maximal) monotone if and only if

$$T^{-1} = \{(y, x) \mid (x, y) \in T\}$$

is (maximal) monotone.

(iii) (*Translation*) Let d be any member of $\mathcal{H}$. Then T is (maximal) monotone if and only if $T + d = \{(x, y+d) \mid (x, y) \in T\}$ is (maximal) monotone.

(iv) (*Linear Change of Coordinates*) Suppose that $\mathcal{H} = \mathbb{R}^n$ and $\mathbf{Q}$ is an invertible $n \times n$ matrix. Then the operator

$$\mathbf{Q}^T T \mathbf{Q} = \{ (\mathbf{Q}^{-1}\mathbf{x}, \mathbf{Q}^T\mathbf{y}) \mid (\mathbf{x}, \mathbf{y}) \in T\}$$

is (maximal) monotone if and only if T is (maximal) monotone Note: when writing a composition of a multivalued operator with a matrix such as $\mathbf{Q}$, $\mathbf{Q}$ should be understood to stand for the equivalent mapping

$$\{(\mathbf{x}, \mathbf{Q}\mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n\}.$$

Proof. (i) and (ii) are immediate. For (iii), it suffices to note that for any $y, y' \in \mathcal{H}$, $(y + d) - (y' + d) = y - y'$. We now prove (iv). Suppose that T is monotone, and consider any two points $(\mathbf{z}, \mathbf{w}), (\mathbf{z}', \mathbf{w}') \in \mathbf{Q}^T T \mathbf{Q}$. By the invertibility of $\mathbf{Q}$ and the definition of $\mathbf{Q}^T T \mathbf{Q}$, there exist unique $(\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}') \in T$ such that $(\mathbf{z}, \mathbf{w}) = (\mathbf{Q}^{-1}\mathbf{x}, \mathbf{Q}^T\mathbf{y})$ and $(\mathbf{z}', \mathbf{w}') = (\mathbf{Q}^{-1}\mathbf{x}', \mathbf{Q}^T\mathbf{y}')$. Then

$$\begin{aligned} \langle \mathbf{z}' - \mathbf{z}, \mathbf{w}' - \mathbf{w} \rangle &= \langle \mathbf{Q}^{-1}(\mathbf{x}' - \mathbf{x}), \mathbf{Q}^T(\mathbf{y}' - \mathbf{y}) \rangle \\ &= (\mathbf{y}' - \mathbf{y})^T \mathbf{Q}\mathbf{Q}^{-1}(\mathbf{x}' - \mathbf{x}) \\ &= \langle \mathbf{x}' - \mathbf{x}, \mathbf{y}' - \mathbf{y} \rangle \\ &\geq 0 \quad . \end{aligned}$$

So, $\mathbf{Q}^T T\mathbf{Q}$ is monotone. If B were a maximal monotone operator strictly containing $\mathbf{Q}^T T\mathbf{Q}$, then

$$(\mathbf{Q}^{-1})^T B\mathbf{Q}^{-1} = \{(\mathbf{Qz}, (\mathbf{Q}^{-1})^T\mathbf{w}) \mid (\mathbf{z}, \mathbf{w}) \in B\}$$

would be a monotone operator strictly containing T. Thus, if T is maximal, so is $\mathbf{Q}^T T\mathbf{Q}$. This proves one direction of (iv). The converse direction follows because

$$T = (\mathbf{Q}^{-1})^T(\mathbf{Q}^T T\mathbf{Q})\mathbf{Q}^{-1} ,$$

as is easily confirmed from the definition of composition. ■

The following two propositions illustrate the role of maximality in the theory of monotone operators. Neither is original (*e.g.* Brézis 1973).

Proposition 3.2. A maximal monotone operator $T: \mathcal{H} \rightrightarrows \mathcal{H}$ is a closed set in $\mathcal{H} \times \mathcal{H}$, that is, if $\{(x_k, y_k)\} \subseteq T$ is a convergent sequence (equivalently, $\{x_k\}$, $\{y_k\} \subseteq \mathcal{H}$ are sequences such that $y_k \in Tx_k$ for all k, and both $\{x_k\}$ and $\{y_k\}$ converge), then

$$\lim_{k \to \infty} (x_k, y_k) \in T ,$$

that is,

$$\lim_{k \to \infty} y_k \in T\left(\lim_{k \to \infty} x_k\right) .$$

Proof. Let $x^\infty = \lim_{k \to \infty} x_k$ and $y^\infty = \lim_{k \to \infty} y_k$. Now consider any $(x, y) \in T$. By monotonicity, $\langle x_k - x, y_k - y\rangle \geq 0$ for all k. By taking limits, we conclude that $\langle x^\infty - x, y^\infty - y\rangle \geq 0$. Since $(x, y) \in T$ was arbitrary, $T \cup \{(x^\infty, y^\infty)\}$ is monotone, and since T is supposed to be maximal, $(x^\infty, y^\infty) \in T$. ■

Proposition 3.3. All maximal monotone operators T are both closed-valued and convex-valued, that is, Tx is a closed convex set for any choice of x.

Proof. We first address the issue of closure. Choose any x, and let $\{y^k\}$ be a convergent sequence in Tx, with limit y^∞. Applying Proposition 3.2 to the sequence $\{(x, y^k)\} \subseteq T$, one obtains that $y^\infty \in Tx$, and so Tx must be closed. We now consider convexity. Fix any x and choose any $y_1, y_2 \in Tx$, and $\lambda \in [0, 1]$. Then for any $(x', y') \in T$, one has

$$\langle x' - x, y' - y_1 \rangle \geq 0$$

$$\langle x' - x, y' - y_2 \rangle \geq 0 \quad ,$$

hence

$$\langle x' - x, y' - (\lambda y_1 + (1 - \lambda) y_2) \rangle \geq 0 \quad .$$

Since $(x', y') \in T$ is arbitrary, one concludes that $T \cup \{(x, \lambda y_1 + (1 - \lambda) y_2)\}$ is monotone, and hence that $\lambda y_1 + (1 - \lambda) y_2 \in Tx$ by the maximality of T. ■

We now present some classical examples of monotone operators from convex analysis. For further details and references, see Rockafellar (1970a). First, some preliminaries:

Definition 3.7. Consider any extended-real-valued function $f: \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$. Then

(i) The (effective) *domain* of f is $\operatorname{dom} f = \{\mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) < \infty\}$.

(ii) f is *proper* if $\operatorname{dom} f \neq \varnothing$.

(iii) The *epigraph* (literally, "above the graph") of f is
$$\operatorname{epi} f = \{(\mathbf{x}, y) \in \mathbb{R}^n \times \mathbb{R} \mid y \geq f(\mathbf{x})\} \quad .$$

(iv) f is said to be *closed* if $\operatorname{epi} f$ is a closed set in $\mathbb{R}^{n+1}$.

(v) f is *lower semicontinuous* if for any sequence $\{\mathbf{x}^k\} \subseteq \mathbb{R}^n$ converging to some $\mathbf{x} \in \mathbb{R}^n$, $f(\mathbf{x}) \leq \liminf_{k \to \infty} f(\mathbf{x}^k)$. This definition is also valid if $\mathbb{R}^n$ is replaced by an arbitrary Hilbert space $\mathcal{H}$.

(vi) f is *polyhedral* if epi f is a polyhedral set.

Theorem 3.1. $f: \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is closed if and only if it is lower semicontinuous.

Definition 3.8. The *subgradient mapping* ∂f of a function $f: \mathcal{H} \to \mathbb{R} \cup \{+\infty\}$ is the operator $\mathcal{H} \rightrightarrows \mathcal{H}$ given by

$$\begin{aligned} \partial f &= \{(x, y) \mid x \in \operatorname{dom} f, y \text{ is a subgradient to } f \text{ at } x\} \\ &= \{(x, y) \mid x \in \operatorname{dom} f, f(x + d) \geq f(x) + \langle d, y \rangle \ \forall\, d \in \mathcal{H}\} \ . \end{aligned}$$

With our existing notation, this definition is equivalent to the usual one (Rockafellar 1966, 1970a), that is, $\partial f(x)$ is the set of subgradients to f at x. We are now ready for a classic result, first proved by Rockafellar (1966). Special cases were previously known to Minty (1962).

Theorem 3.2 (Rockafellar 1966, 1970a). If $f: \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is convex, then $\partial f: \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is a monotone operator. If f is also closed (lower semicontinuous) and proper, ∂f is maximal monotone on $\mathbb{R}^n$.

Note that the converse of Theorem 3.2 does not hold; that is, not all monotone operators on $\mathbb{R}^n$ are subgradients of functions. A stronger form of the theorem is that an operator is the subgradient of a convex function if and only if it is *cyclically monotone* (Rockafellar

1966, 1970a), which is a slightly more stringent requirement. For our purposes, it suffices to note that the (single-valued and linear) operator $\Theta: \mathbb{R}^2 \to \mathbb{R}^2$ given by

$$\Theta \mathbf{x} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \mathbf{x}$$

is monotone, but being given by an asymmetric matrix, is not the subgradient of any function.

There is one special case of the subgradient operator that we will use frequently:

Definition 3.9. The *normal cone operator* of a set $C \subseteq \mathbb{R}^n$ is

$$N_C = \{(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in C, \mathbf{y} \text{ normal to } C \text{ at } \mathbf{x}\}$$

$$= \{(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \in C, \langle \mathbf{x}' - \mathbf{x}, \mathbf{y} \rangle \leq 0 \ \forall \mathbf{x}' \in C\} \quad .$$

The *indicator function* $\delta_C: \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ of a set $C \subseteq \mathbb{R}^n$ is given by

$$\delta_C(\mathbf{x}) = \begin{cases} 0, & \mathbf{x} \in C \\ +\infty, & \mathbf{x} \notin C \end{cases} \quad .$$

Proposition 3.4. For any set $C \subseteq \mathbb{R}^n$, $N_C = \partial(\delta_C)$. If C is convex, then N_C is monotone. If C is nonempty, closed, and convex, then N_C is a maximal monotone operator.

Proof. The first statement follows immediately from the definitions. If C is convex, then δ_C is convex, and the second statement follows from Theorem 3.2. Since $C \neq \varnothing$, δ_C is proper. If C is closed, then

$$\text{epi } \delta_C = C \times [0, \infty)$$

is a closed set, hence δ_C is a proper, closed function, and N_C is maximal. ■

Proposition 3.5. Let V be a linear subspace of $\mathbb{R}^n$, and $V^\perp$ denote its orthogonal complement. For any $\mathbf{d} \in \mathbb{R}^n$, let $V + \mathbf{d}$ denote the affine space $\{\mathbf{v} + \mathbf{d} \mid \mathbf{v} \in V\}$. Then $N_{V+\mathbf{d}}$ is maximal monotone and

$$N_{V+\mathbf{d}} = (V + \mathbf{d}) \times V^\perp \quad .$$

In particular, $N_V = V \times V^\perp$.

Proof. $V + \mathbf{d}$ is closed and convex, so $N_{V+\mathbf{d}}$ is maximal monotone. Take any $\mathbf{x} = \mathbf{v} + \mathbf{d} \in V + \mathbf{d}$ and $\mathbf{w} \in V^\perp$. Then, for any $\mathbf{x}' = \mathbf{v}' + \mathbf{d} \in V + \mathbf{d}$, one has

$$\langle \mathbf{x} - \mathbf{x}', \mathbf{w} \rangle = \langle \mathbf{v} - \mathbf{v}', \mathbf{w} \rangle = 0 \ ,$$

since $\mathbf{v} - \mathbf{v}' \in V$. Since $\mathbf{x}' \in V + \mathbf{d}$ was arbitrary, $(\mathbf{v} + \mathbf{d}, \mathbf{w}) \in N_{V+\mathbf{d}}$. Furthermore, as $\mathbf{x} = \mathbf{v} + \mathbf{d}$ and $\mathbf{w} \in V^\perp$ were arbitrary, it follows that $(V + \mathbf{d}) \times V^\perp \subseteq N_{V+\mathbf{d}}$. To show the opposite conclusion, take any $(\mathbf{x}, \mathbf{y}) \in N_{V+\mathbf{d}}$, and suppose that $\mathbf{y} \notin V^\perp$. Write $\mathbf{y}$ as $\mathbf{u} + \mathbf{w}$, where $\mathbf{u} \in V \setminus \{0\}$ and $\mathbf{w} \in V^\perp$. But $\mathbf{x} \in V + \mathbf{d}$ implies that $\mathbf{x} + \mathbf{u} \in V + \mathbf{d}$, and one obtains

$$\langle (\mathbf{x} + \mathbf{u}) - \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{u}, \mathbf{y} \rangle = \langle \mathbf{u}, \ \mathbf{u} + \mathbf{w} \rangle = \|\mathbf{u}\|^2 > 0 \ ,$$

contradicting the definition of $N_{V+\mathbf{d}}$. ■

Corollary 3.5.1. Finite-dimensional *constant operators* of the form $\{ (\mathbf{x}, \mathbf{a}) \mid \mathbf{x} \in \mathbb{R}^n \}$, where $\mathbf{a} \in \mathbb{R}^n$ is fixed, are maximal monotone.

Proof. By Proposition 3.5, operators of the form

$$(\{0\} + \mathbf{a}) \times \{0\}^\perp = \{ (\mathbf{a}, \mathbf{x}) \mid \mathbf{x} \in \mathbb{R}^n \}$$

are maximal monotone. The result follows by taking inverses. ■

3.2 *The Gradient and Proximal Point Steps; Firmly Nonexpansive Operators*

Consider the problem of minimizing a convex function $f: \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$. From the definition of the subgradient operator ∂f, an equivalent statement of this problem is

$$\text{Find } \mathbf{x} \in \mathbb{R}^n \text{ such that } \mathbf{0} \in \partial f(\mathbf{x}) \ .$$

That is, one seeks a point $\mathbf{x}$ whose image under the monotone operator ∂f contains $\mathbf{0}$. The same problem may be stated for a general monotone operator $T: \mathcal{H} \rightrightarrows \mathcal{H}$, where $\mathcal{H}$ is an arbitrary Hilbert space:

$$\text{Find } x \in \mathcal{H} \text{ such that } 0 \in Tx \ .$$

Definition 3.10. A *zero* of a monotone operator $T: \mathcal{H} \rightrightarrows \mathcal{H}$ is any point $x \in \mathcal{H}$ such that $0 \in Tx$. The set of all zeroes of T, which may be written $T^{-1}(0)$, will also be denoted by "zer T".

Finding a zero is the canonical problem associated with monotone operators. By proposition 3.1(ii), it is equivalent to the seemingly more general problem of finding, given some $d \in \mathcal{H}$, a point $x \in \mathcal{H}$ such that $d \in Tx$.

3.2.1 The Gradient Step

One of the simplest ways to minimize a convex function f is via the *gradient method*, or, in the case that f is not differentiable, the *subgradient method*. For a function f that is finite and differentiable throughout $\mathbb{R}^n$, the gradient method takes the form

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma_k \nabla f(\mathbf{x}^k) \ ,$$

where $\{\gamma_k\}_{k=0}^{\infty} \subseteq (0, \infty)$ is a sequence of *stepsizes*. For a non-differentiable function, this method generalizes to the *subgradient* method,

$$\begin{aligned} &\text{Choose an arbitrary } \mathbf{y}^k \in \partial f(\mathbf{x}^k) \\ &\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma_k \mathbf{y}^k \ , \end{aligned}$$

or, more succinctly,

$$\mathbf{x}^{k+1} \in \mathbf{x}^k - \gamma_k \partial f(\mathbf{x}^k) \ .$$

When f is convex, the gradient and subgradient method may be thought of as methods for finding zeroes of the single-valued cyclically monotone operator ∇f or the multiple-valued cyclically monotone operator ∂f, respectively. For a general monotone operator T, a natural generalization is the method

$$\mathbf{x}^{k+1} \in \mathbf{x}^k - \gamma_k T\mathbf{x}^k \ , \qquad \text{(FSM)}$$

which we call the *forward step method.* This method is illustrated in Figure 1. The following simple proposition partially demonstrates its plausibility.

Proposition 3.6. Let $T: \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ be a monotone operator, and $\{\mathbf{x}^k\}_{k=0}^{\infty} \subseteq \mathbb{R}^n$ and $\{\gamma^k\}_{k=0}^{\infty} \subseteq (0, \infty)$ be sequences conforming to (FSM). If $\{\mathbf{x}^k\}$ converges to some $\mathbf{x} \in \mathbb{R}^n$, there exists some ε such that $0 < \varepsilon \leq \gamma_k$ for all k, and T is maximal, then $\mathbf{x}$ is a zero of T.

Proof. For all k, let $\mathbf{y}^k = (\mathbf{x}^k - \mathbf{x}^{k+1})/\gamma_k \in T\mathbf{x}^k$. Since $\mathbf{x}^k - \mathbf{x}^{k+1} \to \mathbf{0}$ and γ_k is bounded away from zero, $\mathbf{y}^k \to \mathbf{0}$. Consider the sequence $\{(\mathbf{x}^k, \mathbf{y}^k)\}_{k=0}^{\infty} \subseteq T$. Since T is maximal monotone,

$$\lim_{k \to \infty} (\mathbf{x}^k, \mathbf{y}^k) = (\mathbf{x}, \mathbf{0}) \in T$$

by Proposition 3.2. ■

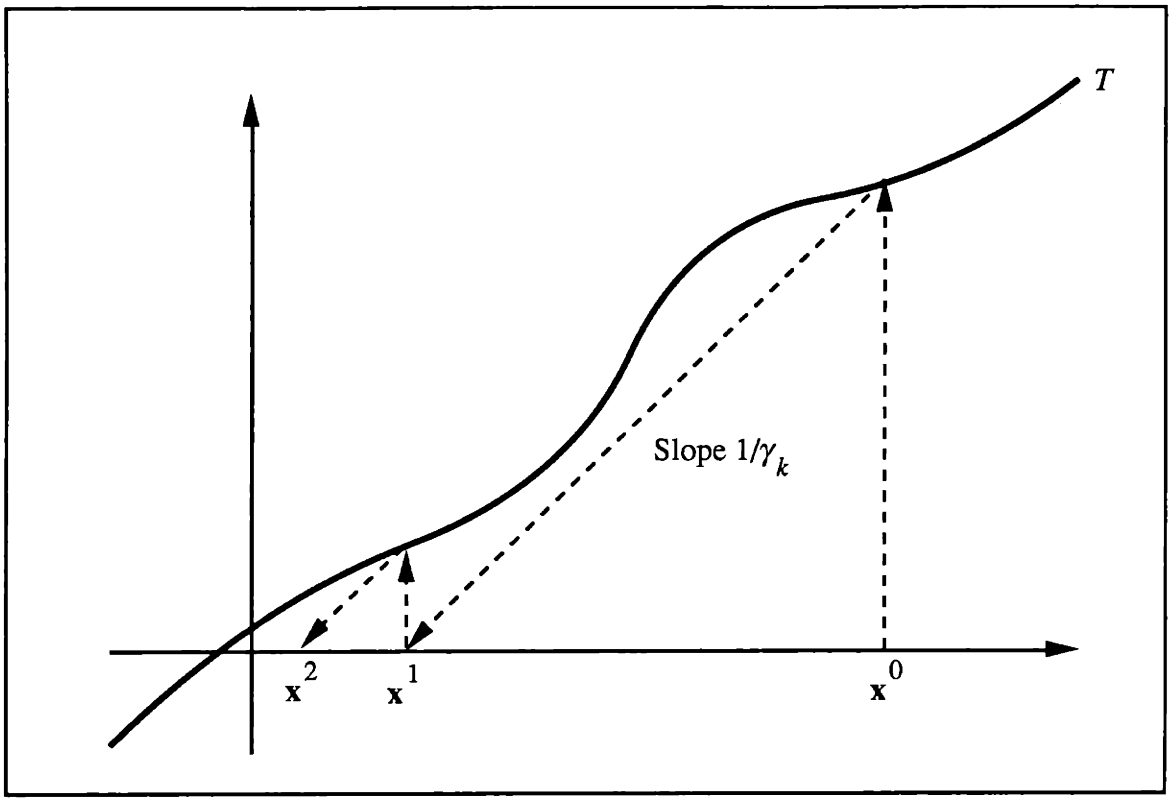


Figure 1. The general forward step method for a monotone operator T.

Unfortunately, the forward step method is guaranteed to converge only under highly restrictive conditions. For the special case of the gradient method, convergence proofs generally assume that the stepsizes γ_k are selected by some rule requiring sampling of the objective function, such as exact line search, Armijo's rule, or Goldstein's rule (*e.g.* Luenberger 1973, Bertsekas 1982). Similarly, in one of the basic convergence results concerning the subgradient method (*e.g.* Shapiro 1979, Polyak 1987), one assumes that the optimal objective value f^* is known, and the the stepsize sequence $\{\gamma_k\}$ meets the condition

$$\varepsilon \frac{f(\mathbf{x}^k) - f^*}{\|\mathbf{y}^k\|^2} \leq \gamma_k \leq (2-\varepsilon) \frac{f(\mathbf{x}^k) - f^*}{\|\mathbf{y}^k\|^2} \qquad \forall\, k \geq 0 \ ,$$

where $\varepsilon \in (0, 1]$, and $\mathbf{y}^k$ denotes the subgradient chosen at step k. Such stepsize rules do not easily extend to the case of general monotone operators, which may lack a corresponding objective function. Bruck (1975a, 1977) has considered a class of stepsize selection rules for general monotone operators, but his results are somewhat difficult to apply in practice.

There is also a body of convergence results for the gradient algorithm without sampling of the objective function, under restrictive assumptions (Goldstein 1964, Levitin and Polyak 1966, *e.g.* Polyak 1987). These results show that if f is convex and ∇f obeys the Lipschitz condition

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')\| \leq L\|\mathbf{x} - \mathbf{x}'\| \qquad \forall \mathbf{x}, \mathbf{x}' \in \operatorname{dom} f \ ,$$

then the gradient method will converge for any stepsize sequence $\{\gamma_k\} \subseteq [\varepsilon, 2L - \varepsilon]$, where $0 < \varepsilon \leq L$ is fixed. It turns out that this result can be adapted to more general monotone operators, so long as they are strongly monotone (as defined below), and remain single-valued. Such extensions abound in the literature of variational inequalities (*e.g.* Sibony 1970, Bakushinski and Polyak 1974, Auslender 1976, Bertsekas and Gafni 1982). We will give a particularly general adaptation due to Gol'shtein and Tret'yakov.

Definition 3.11. An operator T: $\mathcal{H} \rightrightarrows \mathcal{H}$ is called *strongly monotone* if there exists some $\alpha > 0$ such that

$$\langle x' - x, y' - y \rangle \geq \alpha \|x' - x\|^2 \qquad \forall\ (x, y), (x', y') \in T \ .$$

α is called the *modulus* of strong monotonicity. Note that a strongly monotone operator is necessarily monotone. A convex function h is called *strongly convex* if ∂h is strongly monotone.

Typically, results from the variational inequality literature imply that the forward step method can find a zero of a strongly monotone and single-valued operator, so long as the stepsizes are sufficiently small, but do not vanish. The Gol'shtein-Tret'yakov result is slightly different:

Lemma 3.1. If the *inverse* of T: $\mathcal{H} \rightrightarrows \mathcal{H}$ is strongly monotone, that is, for some $\alpha > 0$,

$$\langle x' - x, y' - y \rangle \geq \alpha \| y' - y \|^2 \qquad \forall\ (x, y), (x', y') \in T \ ,$$

then T is single-valued.

Proof. Take any $x \in \mathcal{H}$ and $y, y' \in Tx$. Then

$$0 \ = \ \langle x - x, y' - y \rangle \ \geq \ \alpha \| y' - y \|^2 \quad ,$$

hence $y = y'$. ■

Theorem 3.3 (Gol'shtein 1975, Gol'shtein and Tret'yakov 1975). Let T: $\mathbb{R}^n \rightrightarrows \mathbb{R}^n$ be a maximal monotone operator such that T^{-1} is strongly monotone with modulus α (hence, T is single-valued). Let $\{\mathbf{x}^k\}_{k=0}^{\infty}$, $\{\mathbf{y}^k\}_{k=0}^{\infty} \subseteq \mathbb{R}^n$ and $\{\gamma_k\}_{k=0}^{\infty}$, $\{\varepsilon_k\}_{k=0}^{\infty} \subseteq (0, \infty)$ be sequences such that for all $k \geq 0$,

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma_k \mathbf{y}^k$$

$$\| \mathbf{y}^k - T(\mathbf{x}^k) \| \ \leq \ \varepsilon_k \ .$$

Then if

$$0 \ < \ \inf_{k \geq 0} \gamma_k \ \leq \ \sup_{k \geq 0} \gamma_k \ < \ 2\alpha$$

$$\sum_{k=0}^{\infty} \varepsilon_k \ < \ + \infty \quad ,$$

the sequence $\{\mathbf{x}^k\}$ converges to some $\mathbf{x} \in \text{zer}\, T$, so long as $\text{zer}\, T \neq \emptyset$.

Note that the theorem allows *approximate* values of the operator to be used, so long as the sum of the errors is finite.

There are a number of modifications of the forward step method that have broader convergence properties. Korpelevich's extragradient method (1976) requires only a single-valued monotone operator; no strong monotonicity is needed. Bruck (1974) presented a scheme that requires only maximality, the idea being to replace T by a sequence of strongly monotone approximations. However, the latter result requires a rather intricate choice of stepsizes.

3.2.2 The Path-Following Analogy

We will now concentrate on an alternative to the forward step method called the *proximal point algorithm* (Rockafellar 1976a). To motivate this method, we need to introduce the notion of the gradient/forward step method as a *path-following* algorithm.

First, suppose we are given a convex, everywhere continuously differentiable function $h: \mathbb{R}^n \to \mathbb{R}$, possessing at least one minimizing point. Consider the first-order differential equation system

$$\frac{d\mathbf{x}(t)}{dt} = -\nabla h(\mathbf{x}(t)) \quad . \tag{SD}$$

A solution $\mathbf{x}(t)$ of this system, given some initial condition $\mathbf{x}(0) = \mathbf{x}^0 \in \mathbb{R}^n$, is called a *steepest-descent path* (Botsaris and Jacobson 1976), and we will accordingly call the set of points $\{\mathbf{x}(t) \mid t \geq 0\}$ along such a path a *steepest-descent trajectory*. By the existence

theorem for differential equations, exactly one steepest-descent path starts from any given point. Intuitively, the steepest descent path should lead towards a minimum of h. The following very general theorem of Bruck (1975b) confirms this fact:

Theorem 3.4 (Bruck 1975b). Let $\mathcal{H}$ be any Hilbert space and let ϕ: $\mathcal{H} \rightarrow (-\infty, +\infty]$ be a proper lower semicontinuous convex function with at least one minimizing point. Then for any $x_0 \in$ cl dom ϕ, there exists a *unique* path x:$[0, \infty) \rightarrow \mathcal{H}$ such that

(i) $x(t) \in \text{dom}(\partial\phi)$ for all $t > 0$

(ii) $-\dfrac{dx(t)}{dt} \in \partial\phi(x(t))$ almost everywhere

(iii) $x(0) = x_0$,

and, furthermore, this path has the properties

(a) $x(\cdot)$ is absolutely continuous on $[\delta, \infty)$ for all $\delta > 0$

(b) In the weak topology on $\mathcal{H}$, the limit of $x(t)$ as $t \rightarrow \infty$ exists and is a minimizer of ϕ.

Corollary 3.4.1. Suppose that h: $\mathbb{R}^n \rightarrow \mathbb{R}$ is a continuous, everywhere differentiable convex function, and that a function $\mathbf{x}$:$[0, \infty) \rightarrow \mathbb{R}^n$ solving the initial value problem

$$\frac{d\mathbf{x}(t)}{dt} = -\nabla h(\mathbf{x}(t)) \qquad \mathbf{x}(0) = \mathbf{x}^0 \qquad \text{(SDDE)}$$

exists. Then $\mathbf{x}(t)$ converges to a limit $\mathbf{x}^* = \lim_{t \rightarrow \infty} \mathbf{x}(t)$, which minimizes h over $\mathbb{R}^n$.

Proof. The path $\mathbf{x}$:$[0, \infty) \rightarrow \mathbb{R}^n$ satisfies conditions (i)-(iii) of Bruck's above theorem, and by the uniqueness of the Bruck path must also have properties (a) and (b). Since the

weak and strong topologies are identical in $\mathbb{R}^n$, (b) implies that $\mathbf{x}^*$ exists and minimizes h. ■

Suppose we apply the simplest numerical method for differential equations, the first order forward Euler approximation, to (SDDE). Given $\mathbf{x}^0$, we generate a sequence of iterates $\{\mathbf{x}^k\}$ intended to lie approximately along the steepest descent path by the following procedure: to obtain $\mathbf{x}^{k+1}$, approximate the curve in the neighborhood of $\mathbf{x}^k$ by a straight line of direction $-\nabla h(\mathbf{x}^k)$ and make a step of length $\gamma_k > 0$ in that direction. Algebraically, one has

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \gamma_k \nabla h(\mathbf{x}^k) \quad .$$

This algorithm is precisely the gradient method for unconstrained optimization; hence, the gradient method may be considered an approximation scheme for following a steepest-descent path to an optimum. If the stepsize γ_k is too large, the approximation may be poor, and the gradient method may be unable to negotiate turns in the path, possibly failing to approach the optimal set. If the problem is ill-conditioned, the steepest-descent path may contain very sharp turns, and an extremely small step size will be required to follow it. Such a situation is depicted in Figure 2.

More generally, one may think of forward step method $\mathbf{x}^{k+1} \in \mathbf{x}^k - \gamma_k T\mathbf{x}^k$ as a forward Euler approximation scheme for paths $\mathbf{x}(t)$ satisfying the inclusion condition

$$\mathbf{0} \in \frac{d\mathbf{x}(t)}{dt} + T\mathbf{x}(t) \qquad \forall\, t \geq 0 \quad ,$$

thus making it a path-following method of a sort. In the special case of the subgradient method, where T is the subgradient of a convex function, Bruck's theorem ensures the

existence of some $\mathbf{x}(t)$ satisfying the condition almost everywhere, so the analogy with path-following is better drawn.

As a path-following approximation, the forward step method inherits all the drawbacks of the gradient method. Furthermore, if the choice of $\mathbf{x}^{k+1}$ from the set $\mathbf{x}^k - \gamma_k T\mathbf{x}^k$ is made arbitrarily (as it generally must), then additional inaccuracies may occur at iterates $\mathbf{x}^k$ where T is multiple-valued.

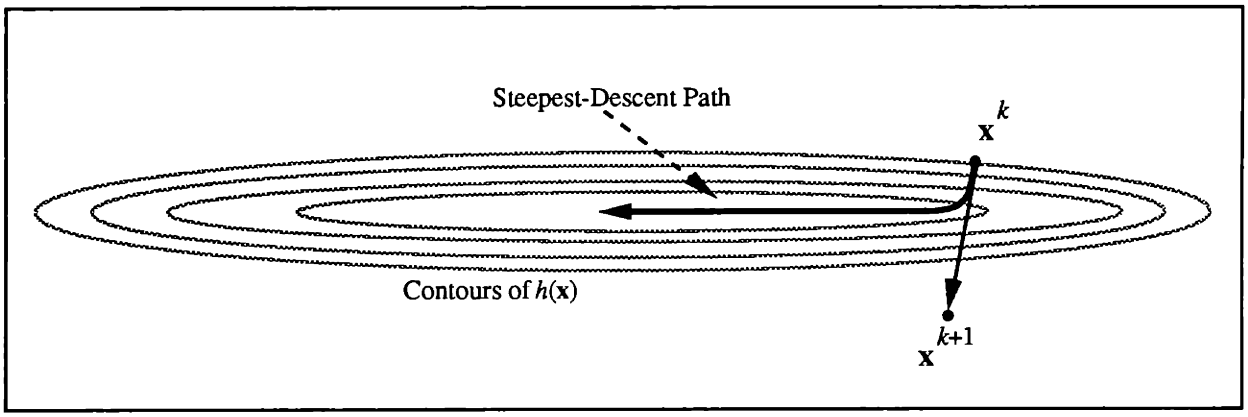


Figure 2. Failure of the gradient method to follow the steepest-descent path in an ill-conditioned problem.

3.2.3 Backward Euler Steps and The Proximal Point Algorithm

Having established the path-following interpretation of the forward step method and its special cases, we now consider an alternate approach: rather than use a forward Euler approximation, why not use the *backward* Euler approximation? The backward Euler method is similar to forward Euler, except that the linear approximation to the path is made around the (unknown) new point $\mathbf{x}^{k+1}$, rather than $\mathbf{x}^k$. In the parlance of numerical

analysis, this kind of calculation is called an *implicit* step, as opposed to the *explicit* step typified by the gradient and forward step methods. Consider first the simple case of minimizing a convex function h that is everywhere defined and differentiable. Then $\mathbf{x}^{k+1}$ is defined as the solution in $\mathbf{x}$ of

$$\mathbf{x} = \mathbf{x}^k - \gamma_k \nabla h(\mathbf{x}) \quad ,$$

or, in functional terms,

$$\mathbf{x}^{k+1} = (I + \gamma_k \nabla h)^{-1}(\mathbf{x}^k) \quad .$$

In the field of numerical methods for differential equations, the backward Euler method is known to have several advantages over forward Euler (*e.g.* Marchuk 1975, Ferziger 1981). In optimization, there are two main advantages to the backward Euler step. First, it turns out that it is not as adversely affected by ill-conditioning as the gradient method, as suggested in Figure 3. Second, it allows much more leeway is selecting stepsizes.

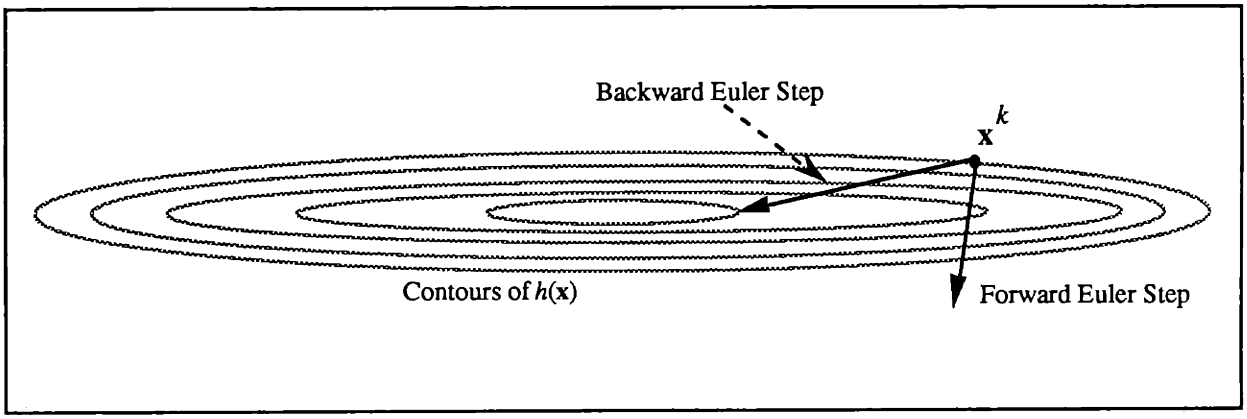


Figure 3. Comparison of the forward and backward Euler steps for an ill-conditioned differentiable minimization problem. The backward step direction is normal to the contour of h at $\mathbf{x}^{k+1}$, whereas the forward step is normal to the contour at $\mathbf{x}^k$.

Now consider the natural generalization of this procedure in which ∇h is replaced by an arbitrary maximal monotone operator T. This iteration may be written, for some sequence $\{\gamma_k\}$ of positive real numbers,

$$\mathbf{x}^{k+1} \in (I + \gamma_k T)^{-1}\mathbf{x}^k \quad , \tag{PP}$$

and is called the *proximal point algorithm* (Rockafellar 1976a). This method is depicted in Figure 4.

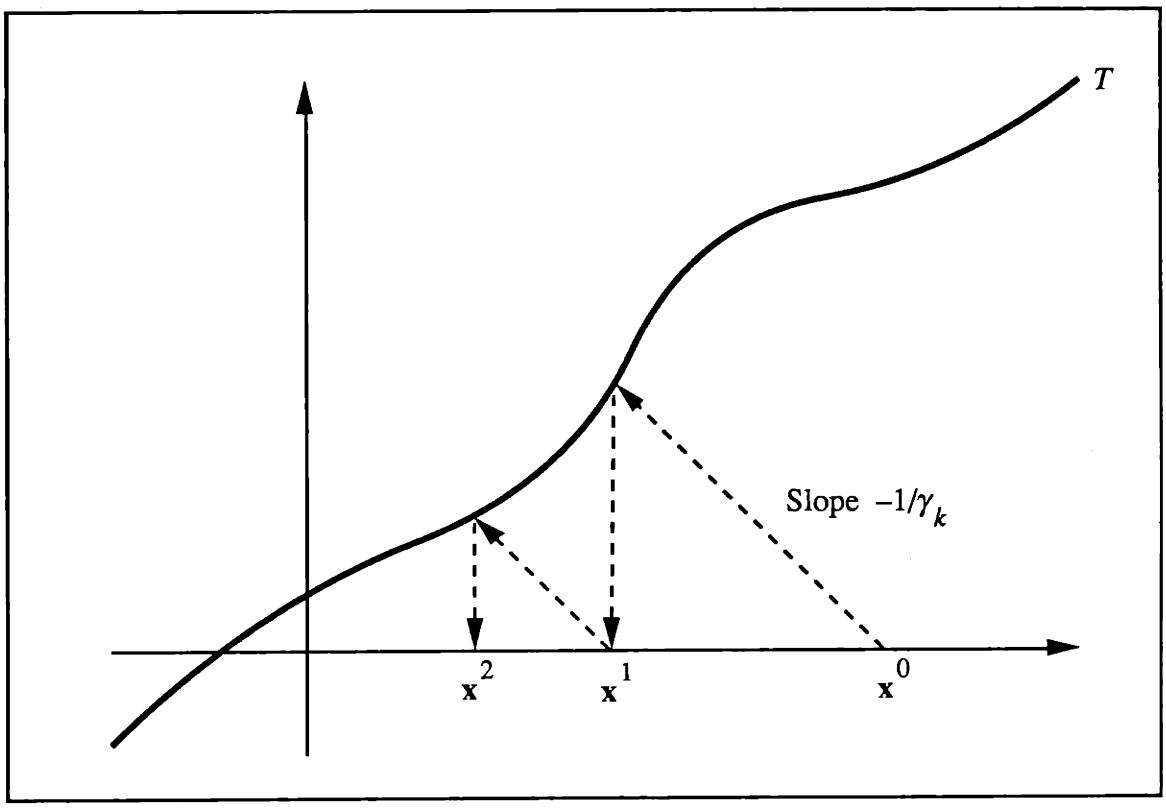


Figure 4. The proximal point algorithm for a monotone operator T.

3.2.4 Resolvents and Firmly Nonexpansive Operators

Before considering the difficulties of actually performing the proximal point iteration (PP), we examine the convergence of the algorithm. To do so, we must consider the properties of mappings of the form $(I + \lambda T)^{-1}$, where T is monotone and λ is a positive scalar. We introduce some relevant notation and terminology.

Definition 3.12. Given any operator $A: \mathcal{H} \rightrightarrows \mathcal{H}$, let J_A denote the operator $(I + A)^{-1}$. For any operator T and positive scalar λ, the operator $J_{\lambda T} = (I + \lambda T)^{-1}$ is called a *resolvent* of T.

The key to the ensuing analysis is that resolvents of monotone operators have a certain *firmly nonexpansive* property, which we will now define.

Definition 3.13. Let $\mathcal{X}$ be any linear space with a norm $\|\cdot\|$. Then a mapping $K: \mathcal{X} \rightrightarrows \mathcal{X}$ is *nonexpansive* if

$$\|y' - y\| \leq \|x' - x\| \quad \forall\ (x, y), (x', y') \in K \ .$$

K is said to be *firmly nonexpansive* if

$$\|y' - y\|^2 \leq \|x' - x\|^2 - \|(x' - y') - (x - y)\|^2 \quad \forall\ (x, y), (x', y') \in K \ .$$

In the literature, operators of the latter type are sometimes called *pseudocontractive* (Reinerman and Schöneberg 1976). The more distinctive term *firmly nonexpansive* appears, for example, in Lions and Mercier (1979).

Proposition 3.7.

(i) All firmly nonexpansive operators are nonexpansive.

(ii) All nonexpansive operators are single-valued and uniformly continuous.

(iii) In a Hilbert space $\mathcal{H}$ with the usual norm $\|x\| = \langle x, x\rangle^{1/2}$, a mapping K is firmly nonexpansive if and only if

$$\|y' - y\|^2 \leq \langle x' - x, y' - y\rangle \quad \forall\ (x, y), (x', y') \in K \ .$$

(iv) All firmly nonexpansive operators are monotone.

Proof. (i) is apparent from the definitions. To prove (ii), let K be nonexpansive and consider any $(x, y_1), (x, y_2) \in K$. Then $0 = \|x - x\| \geq \|y_1 - y_2\|$, hence $y_1 = y_2$, and K must be single-valued. By definition, a nonexpansive operator satisfies a Lipschitz continuity condition, and therefore is uniformly continuous. Now consider (iii). Take any $(x, y), (x', y') \in K$, and note that

$$\begin{aligned} & \|y' - y\|^2 \leq \|x' - x\|^2 - \|(x' - y') - (x - y)\|^2 \\ \Leftrightarrow \quad & \|y' - y\|^2 \leq \|x' - x\|^2 - (\|x' - x\|^2 - 2\langle x' - x, y' - y\rangle + \|y' - y\|^2) \\ \Leftrightarrow \quad & 2\|y' - y\|^2 \leq 2\langle x' - x, y' - y\rangle \ . \end{aligned}$$

This observation establishes (iii), which implies (iv). ■

The following proposition is helpful in visualizing firmly nonexpansive operators in Hilbert space. It is already well known (*e.g.* Rockafellar 1976a).

Proposition 3.8. A single-valued map $J: \mathcal{H} \to \mathcal{H}$, where $\mathcal{H}$ is any Hilbert space, is firmly nonexpansive if and only if $2J - I$ is nonexpansive. Equivalently, a map $\mathcal{H} \to \mathcal{H}$ is firmly nonexpansive if and only if it is of the form $\frac{1}{2}(C + I)$, where C is nonexpansive.

Proof. First, let J be firmly nonexpansive. Then for any $x, y \in \mathcal{H}$,

$$\|(2J - I)x - (2J - I)y\|^2 = 4\|Jx - Jy\|^2 - 4\langle Jx - Jy, x - y\rangle + \|x - y\|^2 \ .$$

Since J is firmly nonexpansive, $4(\|Jx - Jy\|^2 - \langle Jx - Jy, x - y\rangle) \leq 0$, and one deduces that

$$\|(2J - I)x - (2J - I)y\|^2 = \|x - y\|^2 ,$$

and so $2J - I$ is nonexpansive. Conversely, now suppose $C = 2J - I$ is nonexpansive. Then $J = \frac{1}{2}(C + I)$, and for any $x, y \in \mathcal{H}$,

$$\begin{aligned} \|Jx - Jy\|^2 &= \tfrac{1}{4}\|Cx - Cy\|^2 + \tfrac{1}{2}\langle Cx - Cy, x - y\rangle + \tfrac{1}{4}\|x - y\|^2 \\ &\leq \tfrac{1}{2}\|x - y\|^2 + \tfrac{1}{2}\langle Cx - Cy, x - y\rangle \\ &= \langle \tfrac{1}{2}(C + I)x - \tfrac{1}{2}(C + I)y, x - y\rangle \\ &= \langle Jx - Jy, x - y\rangle . \end{aligned}$$

Therefore, J is firmly nonexpansive. This proves the first claim; the second claim is simply a reformulation of the first. ■

Figure 5 gives a characterization of firmly nonexpansive operators that follows directly from Proposition 3.8.

We now demonstrate the connection between monotonicity, maximality and firm nonexpansiveness. First, we need one additional result.

Theorem 3.5. Given a monotone operator T on a Hilbert space $\mathcal{H}$, T is maximal if and only if $\text{im}(I+T) = \mathcal{H}$.

The proof of this theorem is deceptively difficult, and involves Zorn's lemma, and hence the axiom of choice. One method of proof is in Minty (1962), while another, leading to somewhat more general conclusions, may be found in Brézis (1973), Doležal (1979), or

Joshi and Bose (1985), with supporting material in Rockafellar (1969), Debrunner and Flor (1964), Fan (1952), and Glicksberg (1952). Unfortunately, both approaches to the proof are too involved to include here.

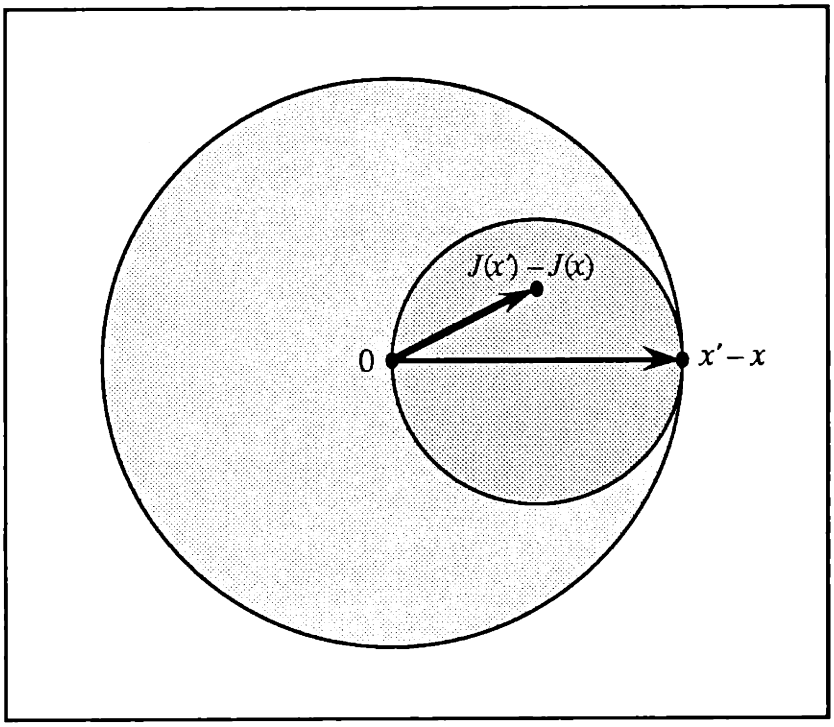


Figure 5. Illustration of the action of firmly nonexpansive operators in Hilbert space. If J is nonexpansive, then $J(x') - J(x)$ must lie in the larger sphere, which has radius $\|x' - x\|$ and is centered at 0. If J is *firmly* nonexpansive, then $J(x') - J(x)$ must lie in the smaller sphere, which has radius $(1/2)\|x' - x\|$ and is centered at $(1/2)(x' - x)$. This characterization follows directly from J being of the form $(1/2)I + (1/2)C$, where C is nonexpansive.

It is now possible to state our key result. The "only if" part of the following theorem has been well known for some time, but the "if" part, just as easily obtained, appears to have been obscure or unknown. The purpose here is to stress the complete symmetry that exists between (maximal) monotone operators and (full-domained) firmly nonexpansive operators over any Hilbert space.

In the past, Rockafellar (1976a) has suggested that the class of firmly nonexpansive operators and the class of resolvents of monotone operators might not be equal; we will now demonstrate that they are identical.

Theorem 3.6. Let $\mathcal{H}$ be a real Hilbert space, and λ any positive scalar. An operator T on $\mathcal{H}$ is monotone if and only if its resolvent $J_{\lambda T} = (I + \lambda T)^{-1}$ is firmly nonexpansive. Furthermore, T is maximal monotone if and only if $J_{\lambda T}$ is firmly nonexpansive and $\mathrm{dom}(J_{\lambda T}) = \mathcal{H}$.

Proof. First, consider the case $\lambda = 1$. By the definition of the addition and inversion operations,

$$(x, y) \in T \Leftrightarrow (x + y, x) \in (I+T)^{-1} \ .$$

Therefore,

$$\begin{aligned} T \text{ monotone} &\Leftrightarrow \langle x' - x, y' - y \rangle \geq 0 && \forall\ (x, y), (x', y') \in T \\ &\Leftrightarrow \langle x' - x + y' - y, x' - x \rangle \geq \|x' - x\|^2 && \forall\ (x, y), (x', y') \in T \\ &\Leftrightarrow (I + T)^{-1} \text{ firmly nonexpansive.} \end{aligned}$$

The first claim is established. By Theorem 3.5, T is maximal if and only if $\mathrm{im}(I+T) = \mathcal{H}$. This is in turn true if and only if the operator $(I+T)^{-1}$ has domain $\mathcal{H}$. This establishes the second statement. The proof for $\lambda \neq 1$ follows from the $\lambda = 1$ case by Proposition 3.1(i). ■

Corollary 3.6.1. An operator K is firmly nonexpansive if and only if $K^{-1} - I$ is monotone. K is firmly nonexpansive with full domain if and only if $K^{-1} - I$ is maximal monotone.

Corollary 3.6.2. For any $\lambda > 0$, the resolvent $J_{\lambda T}$ of a monotone operator T is single-valued. If T is also maximal, then $J_{\lambda T}$ has full domain.

Corollary 3.6.3 (The Representation Lemma). Let $\lambda > 0$ be fixed and T: $\mathcal{H} \rightrightarrows \mathcal{H}$ be monotone. Then every element z of $\mathcal{H}$ can be written in at most one way as $x + \lambda y$, where $y \in Tx$. If T is maximal, then every element z of $\mathcal{H}$ can be written in *exactly* one way as $x + \lambda y$, where $y \in Tx$.

Corollary 3.6.4. The functional taking each operator T to $(I+T)^{-1}$ is a bijection between the collection of maximal monotone operators on $\mathcal{H}$ and the collection of firmly nonexpansive operators on $\mathcal{H}$.

Corollary 3.6.1 is simply a restatement of the $\lambda = 1$ case of the theorem, while Corollary 3.6.2 follows directly from the theorem and Proposition 3.7(ii). Corollary 3.6.3 is essentially a restatement of 3.6.2. Corollary 3.6.4 resembles a result of Minty (1962), but is not identical (Minty did not use the concept of *firm* nonexpansiveness).

It also turns out that the zeroes of a monotone operator precisely coincide with the fixed points of its resolvents:

Proposition 3.9. Given any maximal monotone operator T: $\mathcal{H} \rightrightarrows \mathcal{H}$, real number $\lambda > 0$, and $x \in \mathcal{H}$, $0 \in Tx$ if and only if $J_{\lambda T}(x) = \{x\}$.

Proof. By direct calculation, $J_{\lambda T} = \{(x + \lambda y, x) \mid (x, y) \in T\}$. Hence,

$$0 \in Tx \quad \Leftrightarrow \quad (x, 0) \in T \quad \Leftrightarrow \quad (x, x) \in J_{\lambda T} \ .$$

Since $J_{\lambda T}$ is single-valued, the proof is complete. ■

3.2.5 *Convergence Results for the Proximal Point Algorithm*

Proposition 3.9 result suggests applying the recursion $x^{k+1} = J_{\lambda T}x^k$ to find a fixed point $J_{\lambda T}$, and hence a zero of T. This procedure is precisely the proximal point algorithm for T with the stepsizes γ_k fixed at λ. Its validity is subsumed in the theorems below, which rely centrally upon the firmly nonexpansive properties of resolvents. Our first result is meant to illustrate the fundamental reasoning involved.

Theorem 3.7. Let $T: \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ be a maximal monotone operator, and $\{\gamma_k\}_{k=0}^{\infty}$ be a sequence of positive scalars. Then for any $\mathbf{x}^0 \in \mathbb{R}^n$, there exists a unique sequence $\{\mathbf{x}^k\}_{k=0}^{\infty}$ satisfying the proximal point recursion $\mathbf{x}^{k+1} \in (I + \gamma_k T)^{-1}\mathbf{x}^k$ for all $k \geq 0$. If in addition one has $\sum_{k=0}^{\infty} \gamma_k^2 = \infty$, and T possesses at least one zero, then $\{\mathbf{x}^k\}$ converges to a zero of T.

Proof. Because T is maximal monotone, the operators $(I + \gamma_k T)^{-1}$ are single-valued and defined everywhere for each k, proving the existence and uniqueness of $\{\mathbf{x}^k\}$. Let $\mathbf{x}^*$ be any zero of T. Then $\mathbf{x}^*$ is a fixed point of $(I + \gamma_k T)^{-1}$ for all k, and by firm nonexpansiveness of these same operators,

$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \leq \|\mathbf{x}^k - \mathbf{x}^*\|^2 - \|\mathbf{x}^{k+1} - \mathbf{x}^k\|^2 \quad \forall\, k \geq 0\,. \tag{B}$$

It follows that if any $\mathbf{x}^*$ exists, $\{\mathbf{x}^k\}$ is bounded and possesses a limit point $\mathbf{x}^\infty$. Because $\mathbf{x}^{k+1} \in (I + \gamma_k T)^{-1}\mathbf{x}^k$,

$$\mathbf{x}^k \in \mathbf{x}^{k+1} + \gamma_k T\mathbf{x}^{k+1}$$

$$\Rightarrow \quad \mathbf{x}^k - \mathbf{x}^{k+1} \in \gamma_k T\mathbf{x}^{k+1} \quad .$$

Let $\mathbf{y}^{k+1} = (1/\gamma_k)(\mathbf{x}^k - \mathbf{x}^{k+1}) \in T\mathbf{x}^{k+1}$ for all $k \geq 0$. Then for $k \geq 1$, the monotonicity of T gives that

$$0 \leq \langle \mathbf{x}^k - \mathbf{x}^{k+1}, \mathbf{y}^k - \mathbf{y}^{k+1} \rangle = \gamma_k \langle \mathbf{y}^{k+1}, \mathbf{y}^k - \mathbf{y}^{k+1} \rangle ,$$

and hence

$$\|\mathbf{y}^{k+1}\|^2 \leq \langle \mathbf{y}^{k+1}, \mathbf{y}^k \rangle \Rightarrow \|\mathbf{y}^{k+1}\| \leq \|\mathbf{y}^k\|$$

by the Cauchy-Schwartz inequality. Thus, the sequence $\{\|\mathbf{y}^k\|\}_{k=1}^{\infty}$ is nonincreasing and must converge to some nonnegative limit. By adding up (B) for $k = 0$ to $N - 1$ and substituting the definition of the $\{\mathbf{y}^k\}$, we obtain

$$\|\mathbf{x}^N - \mathbf{x}^*\|^2 \leq \|\mathbf{x}^0 - \mathbf{x}^*\|^2 - \sum_{k=0}^{N-1} \gamma_k^2 \|\mathbf{y}^{k+1}\|^2 \quad \forall\, N \geq 1 .$$

Since $\|\mathbf{x}^0 - \mathbf{x}^*\|^2$ is finite and $\sum_{k=0}^{\infty} \gamma_k^2$ is not, we conclude by letting $N \to \infty$ that $\|\mathbf{y}^{k+1}\|^2 \to 0$, hence $\mathbf{y}^k \to \mathbf{0}$. Now consider any subsequence $\{\mathbf{x}^{k(j)}\}_{j=0}^{\infty}$ of $\{\mathbf{x}^k\}$ converging to the limit point $\mathbf{x}^{\infty}$. We have $\{(\mathbf{x}^{k(j)}, \mathbf{y}^{k(j)})\} \subseteq T$ and convergent to $(\mathbf{x}^{\infty}, \mathbf{0})$, so $\mathbf{0} \in T\mathbf{x}^{\infty}$ by Proposition 3.2. Substituting $\mathbf{x}^{\infty}$ for $\mathbf{x}^*$ in (B), one obtains

$$\|\mathbf{x}^{k+1} - \mathbf{x}^{\infty}\|^2 \leq \|\mathbf{x}^k - \mathbf{x}^{\infty}\|^2 \quad \forall\, k \geq 0 ,$$

so $\{\mathbf{x}^k\}$ cannot have any other limit points besides $\mathbf{x}^{\infty}$, and must converge to $\mathbf{x}^{\infty}$. ■

The main argument of Theorem 3.7 follows that of Brézis and Lions (1978), whose result extends to weak convergence in infinite-dimensional Hilbert spaces by appealing to a result of Opial (1967). In general Hilbert space, the proof given above is sufficient to show that $\{\mathbf{x}^k\}$ has a weak limit point; Opial's result is needed to show that the sequence has no *other* weak limit point. The key observation required is that $\{\|\mathbf{x}^k - \mathbf{x}^*\|\}$ is nonincreasing for all $\mathbf{x}^* \in \operatorname{zer} T$.

We will also give, without proof, proximal point convergence theorems by Rockafellar (1976a) and Gol'shtein and Tret'yakov (1979). The Rockafellar result predates Brézis

and Lions', and was the first general proximal point convergence theorem — special cases were proved earlier by Martinet (1970, 1972). The Rockafellar proof is slightly more restrictive as to stepsizes, but provides two additional benefits: it allows the resolvent to be evaluated approximately, and it indicates what happens if T possesses no zeroes.

Theorem 3.8 (Rockafellar 1976a). Let T: $\mathcal{H} \rightrightarrows \mathcal{H}$ be a maximal monotone operator on an arbitrary Hilbert space $\mathcal{H}$, and x^0 be any element of $\mathcal{H}$. Let $\{\gamma_k\}_{k=0}^{\infty} \subseteq (0, \infty)$, $\{\varepsilon_k\}_{k=0}^{\infty} \subseteq [0, \infty)$, and $\{x^k\}_{k=0}^{\infty} \subseteq \mathcal{H}$ be sequences satisfying the conditions

$$\|x^{k+1} - (I + \gamma_k T)^{-1} x^k\| \leq \varepsilon_k \quad \forall\, k \geq 0$$

$$\inf_{k \geq 0} \{\gamma_k\} > 0$$

$$\sum_{k=0}^{\infty} \varepsilon_k < \infty \ .$$

If $\mathrm{zer}(T) \neq \varnothing$, $\{x^k\}$converges in the weak topology to some zero of T. If $\mathrm{zer}(T) = \varnothing$, then $\|x^k\| \to \infty$.

The following alternative theorem may be deduced directly from material in Gol'shtein (1975) and Gol'shtein and Tret'yakov (1979). Here, the stepsize is held constant, but *under-* or *over-relaxation factors* $\{\rho_k\}_{k=0}^{\infty}$ are introduced. While the possibility of approximate computation is retained, no claim is made for the infeasible or infinite-dimensional cases.

Theorem 3.9. (Gol'shtein and Tret'yakov 1979) Let T: $\mathbb{R}^n \rightrightarrows \mathbb{R}^n$ be a maximal monotone operator, and γ be a positive scalar. Let $\{\rho_k\}_{k=0}^{\infty} \subseteq (0, 2)$, $\{\varepsilon_k\}_{k=0}^{\infty} \subseteq [0, \infty)$, $\{x^k\}_{k=0}^{\infty} \subseteq \mathbb{R}^n$, and $\{y^k\}_{k=0}^{\infty} \subseteq \mathbb{R}^n$ be sequences satisfying the conditions

$$0 < \inf_{k \geq 0} \rho_k \leq \sup_{k \geq 0} \rho_k < 2$$

$$\sum_{k=0}^{\infty} \varepsilon_k < +\infty \quad ,$$

$$\|y^k - (I + \gamma T)^{-1} x^k\| \leq \varepsilon_k \qquad \forall\, k \geq 0$$

$$x^{k+1} = (1 - \rho_k) x^k + \rho_k y^k \qquad \forall\, k \geq 0 \quad .$$

Then if T has any zeroes, $\{x^k\}$ converges to one of them.

The procedure by which Gol'shtein and Tret'yakov establish the above result is quite interesting: they first convert the algorithm to a *forward* step method on a *modified* monotone operator satisfying the inverse strong monotonicity condition of Lemma 3.1, and then apply Theorem 3.3. Thus, they show that the forward and fixed-stepsize backward step methods are in some sense the same. However, to evaluate the modified operator, one must essentially perform the proximal point iteration. Therefore, both to gain intuition about the proximal point algorithm and to permit its analysis with varying stepsizes, it still seems desirable to preserve the conceptual distinction between the forward and backward steps.

Comparing convergence results for the forward step and proximal point algorithms, one notes that the proximal point algorithm has a number of advantages. First, it converges to a zero of a maximal monotone operator under only mild conditions on the stepsize sequence, whereas the forward step method requires both special problem structure and a more restrictive choice of stepsizes. Second, for some monotone operators T, there is a danger that the forward step method may "jam" by stepping to a point x^k where $Tx^k = \varnothing$. Such a difficulty cannot occur in the proximal point algorithm so long as T is maximal.

The principal difficulty in applying the proximal point algorithm is that the proximal point iteration $\mathbf{x}^{k+1} = (I + \lambda_k T)^{-1}\mathbf{x}^k$ may be much more difficult to perform than the corresponding forward step iteration. Fundamentally, this is because the proximal point iteration involves *inverting* a known operator, while the forward step calculation does not. For example, in trying to solve the problem

$$\text{minimize}_{\ \mathbf{x}\in\mathbb{R}^n}\ h(\mathbf{x})\ ,$$

where h is closed proper convex, one might attempt to apply the proximal point algorithm to the operator ∂h. This approach is called the *proximal minimization algorithm* (Rockafellar 1976b). Unfortunately, as will be shown later in this chapter, finding $\mathbf{x}^{k+1} = (I + \lambda_k \partial h)^{-1}\mathbf{x}^k$ turns out to be equivalent to computing

$$\mathbf{x}^{k+1} = \underset{\mathbf{x}\in\mathbb{R}^n}{\arg\min}\left\{h(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{x}-\mathbf{x}^k\|^2\right\}\ ,$$

and so each iteration of the proximal point algorithm involves a calculation that is, in general, as hard as the entire original problem (although the presence of the quadratic term, guaranteeing that the minimand is strongly convex, may be an advantage in some contexts).

However, there do exist cases in which the proximal point iteration is not much more difficult than the forward step method, or perhaps easier. In such cases, the proximal point iteration is to be preferred. A notable case is when the operator T is the subdifferential of a dual functional generated by some convex program, and for which no explicit formula is available. This situation occurs in the *method of multipliers*, which Rockafellar (1976b) has shown to be an application of the proximal point algorithm to a particular dual functional. The forward step method for the same functional yields the subgradient method for Lagrangian relaxation (as applied to convex programming). The method of

multipliers has several advantages over subgradient optimization of Lagrangians: one need not know the optimal objective value to be assured of convergence, and both primal and dual solutions converge to respective optima, whereas in Lagrangian relaxation the primal solution sequence may not converge, and can have infeasible limit points.

Still, because backward step schemes are generally more difficult to implement, forward step methods and their refinements are much more prevalent in mathematical programming. Typical refinements include line search to determine the step size, and use of higher order derivative information. The well-known Newton approach uses second derivative information to apply a coordinate transformation that locally straightens the steepest-descent path, thus improving the accuracy of the forward step. Botsaris and Jacobson (1976, also see Botsaris 1978a, 1978b) have proposed methods that search along higher-order — as opposed to linear — approximations of the steepest-descent path. Such refinements do not easily translate themselves into the general domain of monotone operators.

This concludes our basic discussion of the basic forward step and proximal point algorithms. Our next main task will be to construct *splitting* algorithms that use these two iterations as building blocks for more complex calculations. But first, we give a few additional results for firmly nonexpansive operators.

3.3 Some Observations about Firmly Nonexpansive Mappings

This section contains some ancillary material about firmly nonexpansive operators that is not absolutely essential to our principal analysis. However, given the importance firmly

nonexpansive operators demonstrated in the previous section, it seems worthwhile to include these results, all of which appear to be original.

The first two observations concern *compositions* of firmly nonexpansive mappings. Such compositions occur fairly frequently in algorithms, so it is helpful to understand their properties. The first result is of a negative nature, showing that compositions of firmly nonexpansive maps need not themselves be firmly nonexpansive.

Proposition 3.10. There exist, even in finite-dimensional Hilbert spaces, firmly nonexpansive operators V and W such that $V \circ W$ is not firmly nonexpansive.

Proof. Consider the space $\mathbb{R}^2$ under the usual inner product. The example we will show uses exclusively linear operators, so we will use matrix and operator notation interchangeably. Let

$$\mathbf{V} = \begin{bmatrix} \frac{2}{3} & -\frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix} .$$

Then

$$2\mathbf{V} - \mathbf{I} = \begin{bmatrix} \frac{1}{3} & -\frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} \end{bmatrix} .$$

For any $\mathbf{x} \in \mathbb{R}^2$, direct calculation shows that

$$\|(2\mathbf{V} - \mathbf{I})\mathbf{x}\| = \frac{\sqrt{5}}{3}\|\mathbf{x}\| \leq \|\mathbf{x}\| ,$$

so $2\mathbf{V} - \mathbf{I}$ is nonexpansive, hence $\mathbf{V}$ (considered as a linear map) is firmly nonexpansive. Letting

$$\mathbf{W} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} \end{bmatrix} ,$$

one has

$$2\mathbf{W} - \mathbf{I} = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{bmatrix} ,$$

and for any $\mathbf{x} \in \mathbb{R}^2$, another direct calculation shows that

$$\|(2\mathbf{W} - \mathbf{I})\mathbf{x}\| = \frac{\sqrt{5}}{3}\|\mathbf{x}\| \leq \|\mathbf{x}\| ,$$

and $\mathbf{W}$ is also firmly nonexpansive. However,

$$\mathbf{VW} = \begin{bmatrix} \frac{1}{3} & \frac{1}{9} \\ \frac{4}{9} & \frac{1}{3} \end{bmatrix} .$$

One may confirm by direct calculation that $\mathbf{VW}$ violates the firm nonexpansiveness condition $\langle \mathbf{x}' - \mathbf{x}, \mathbf{y}' - \mathbf{y} \rangle \geq \|\mathbf{y}' - \mathbf{y}\|^2$ for the two vectors $\mathbf{x}' = (-(2 + \sqrt{2}), 1)$ and $\mathbf{x} = (0, 0)$. ■

Despite this negative result, it turns out that compositions of firmly nonexpansive operators still retain some of the convergence properties of truly firmly nonexpansive operators, as we now show. This result also gives some insight as to why the proximal point algorithm converges when the stepsize is allowed to vary.

Theorem 3.10. Suppose that $R_1,...,R_p: \mathbb{R}^n \to \mathbb{R}^n$ are firmly nonexpansive (hence single-valued) operators such that $S = R_p \circ R_{p-1} \circ ... \circ R_1$ possesses a fixed point, and that the sequence $\{\mathbf{x}^k\}$ obeys the recursion $\mathbf{x}^{k+1} = S\mathbf{x}^k$ for some $\mathbf{x}^0 \in \mathbb{R}^n$. Then $\{\mathbf{x}^k\}$ converges to

a fixed point of S.

Proof. Let S_0 be the identity map, and for $q=1,...,p$, let

$$S_q = R_q \circ R_{q-1} \circ \ldots \circ R_1 ,$$

so that $S_p = S$ and $\mathbf{x}^{k+1} = S_p \mathbf{x}^k$. Let $\mathbf{x}^*$ be any fixed point of S_p. By firm nonexpansiveness, for any $q=1,...,p$ and all $k \geq 0$,

$$\|R_q S_{q-1}\mathbf{x}^k - R_q S_{q-1}\mathbf{x}^*\|^2 \leq \|S_{q-1}\mathbf{x}^k - S_{q-1}\mathbf{x}^*\|^2 - \|(S_{q-1}-R_q S_{q-1})\mathbf{x}^k - (S_{q-1}-R_q S_{q-1})\mathbf{x}^*\|^2 ,$$

or

$$\|S_q\mathbf{x}^k - S_q\mathbf{x}^*\|^2 \leq \|S_{q-1}\mathbf{x}^k - S_{q-1}\mathbf{x}^*\|^2 - \|(S_{q-1} - S_q)\mathbf{x}^k - (S_{q-1} - S_q)\mathbf{x}^*\|^2 .$$

Combining these inequalities for all q, one obtains

$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \leq \|\mathbf{x}^k - \mathbf{x}^*\|^2 - \sum_{q=1}^{p} \|(S_{q-1} - S_q)\mathbf{x}^k - (S_{q-1} - S_q)\mathbf{x}^*\|^2 . \qquad (*)$$

Therefore, $\{\mathbf{x}^k\}$ is bounded and thus possesses some limit point $\mathbf{x}^\infty$. Furthermore, by using the above inequality for $k=0,...,K-1$, we further obtain that

$$\|\mathbf{x}^k - \mathbf{x}^*\|^2 \leq \|\mathbf{x}^0 - \mathbf{x}^*\|^2 - \sum_{q=1}^{p} \left(\sum_{k=0}^{K-1} \|(S_{q-1} - S_q)\mathbf{x}^k - (S_{q-1} - S_q)\mathbf{x}^*\|^2 \right) .$$

Since $\|\mathbf{x}^k - \mathbf{x}^*\|^2 \geq 0$, it follows by letting $K \to \infty$ that for $q=1,...,p$,

$$\sum_{k=0}^{\infty} \|(S_{q-1} - S_q)\mathbf{x}^k - (S_{q-1} - S_q)\mathbf{x}^*\|^2 \leq \infty$$

$$\Rightarrow \lim_{k \to \infty} \|(S_{q-1} - S_q)\mathbf{x}^k - (S_{q-1} - S_q)\mathbf{x}^*\|^2 = 0 .$$

Therefore,

$$\lim_{k \to \infty} \left(S_{q-1}\mathbf{x}^k - S_q\mathbf{x}^k \right) = S_{q-1}\mathbf{x}^* - S_q\mathbf{x}^* \qquad \forall\, 1 \le q \le p \quad .$$

Adding these equations together gives

$$\lim_{k \to \infty} \left(\mathbf{x}^k - S_p\mathbf{x}^k \right) = \mathbf{x}^* - S_p\mathbf{x}^* = 0 \quad .$$

Let $\{\mathbf{x}^{k(j)}\}$, j=1,2,... , be a subsequence of $\{\mathbf{x}^k\}$ converging to the limit point $\mathbf{x}^\infty$. From the above, we know that

$$\lim_{j \to \infty} S_p\mathbf{x}^{k(j)} = \lim_{j \to \infty} \mathbf{x}^{k(j)} = \mathbf{x}^\infty \quad .$$

Since the $R_1,...,R_p$ are continuous, their composition S_p is continuous, and so

$$S_p\mathbf{x}^\infty = \lim_{j \to \infty} S_p\mathbf{x}^{k(j)} = \mathbf{x}^\infty \quad ,$$

and so $\mathbf{x}^\infty$ is a fixed point of S_p. By substituting $\mathbf{x}^\infty$ for $\mathbf{x}^*$ in (*), it is easily demonstrated that $\mathbf{x}^\infty$ is the only limit point of $\{\mathbf{x}^k\}$. Since $\{\mathbf{x}^k\}$ is bounded, the proof is complete. ■

Our last results illuminate the connection between firmly nonexpansive operators and projection operators. For any Hilbert space $\mathcal{H}$, and nonempty closed convex set $C \subseteq \mathcal{H}$, let P_C denote the (single-valued and everywhere defined) operator of projection onto C.

Proposition 3.11. Let $\mathcal{H}$ be any Hilbert space.

(i) All projection operators for closed convex subsets of $\mathcal{H}$ are firmly nonexpansive; in fact $P_C = (I + N_C)^{-1}$ for all nonempty closed convex sets $C \subseteq \mathcal{H}$.

(ii) An operator P on $\mathcal{H}$ is the projection operator for some nonempty, closed, convex set C if and only if it is firmly nonexpansive, defined

everywhere, and idempotent, that is, $P \circ P = P$. More specifically, if P has these properties, then the set $C = \{x \in \mathcal{H} \mid Px = x\}$ is closed, nonempty, and convex, and $P = P_C$.

Proof. (i) Let $C \neq \emptyset$ be closed and convex. Then N_C is maximal monotone and $(I + N_C)^{-1}$ is firmly nonexpansive with domain $\mathcal{H}$. Let x be any point in $\mathcal{H}$. Then

$$y = (I + N_C)^{-1}x \iff (I + N_C)y \ni x \iff x - y \in N_C y \quad ,$$

that is, $y = (I + N_C)^{-1}x$ if and only if $y \in C$ and $x - y$ is in the normal cone to C at y, which means y is the projection of x onto C. Hence, $P_C = (I + N_C)^{-1}$ and (i) is proven. We now establish (ii). Given any $C \neq \emptyset$, $P_C \circ P_C = P_C$, so the "only if" statement holds by part (i). Now let P be an idempotent, everywhere-defined, firmly nonexpansive map, and define $C = \{x \in \mathcal{H} \mid Px = x\}$ as above. Taking any $x \in \mathcal{H}$, we have $PPx = Px$, hence $Px \in C$, and C is nonempty. Now let $\{x^k\}$ be any convergent sequence in C, with limit x^∞. P is nonexpansive, hence continuous, so it follows

$$Px^\infty = P\left(\lim_{k \to \infty} x^k\right) = \lim_{k \to \infty} Px^k = \lim_{k \to \infty} x^k = x^\infty \quad ,$$

so x^∞ must be in C, and C is closed. To show that C is convex, consider the operator $P^{-1} - I$, which is maximal monotone by Corollary 3.6.1. Then

$$x \in C \iff x = Px \iff P^{-1}x - x \ni 0 \iff x \in (P^{-1} - I)^{-1}(0) \quad .$$

From Proposition 3.1(ii), $(P^{-1} - I)^{-1}$ is maximal monotone, and so by Proposition 3.3, $C = (P^{-1} - I)^{-1}(0)$ is a convex set. Finally, consider any $x \in \mathcal{H}$. By firm nonexpansiveness,

$$\langle x - w, Px - w \rangle \geq \|Px - w\|^2 = \langle Px - w, Px - w \rangle \quad \forall\, w \in C \quad ,$$

whence

$$\langle x - Px, Px - w \rangle \geq 0 \quad \forall\, w \in C \ .$$

By the idempotence of P, $Px \in C$, hence Px is the projection of x onto C, proving the remainder of (ii). ■

We should note that part (ii) of the above proposition appears to be original.

Corollary 3.11.1. There exist firmly nonexpansive, everywhere-defined maps that are not projections.

Proof. Take any $d \in \mathcal{H}$, $d \neq 0$. Then the map $J(x) = x + d$ is firmly nonexpansive, yet not idempotent. ■

Corollary 3.11.2. For any nonempty, closed, convex set C and $\gamma > 0$, $(I + \gamma N_C)^{-1}x$ is the projection of x onto C.

Proof. It suffices to note that $\gamma N_C = N_C$ for any $\gamma > 0$. ■

Corollary 3.11.3. Let $C_1,...,C_p \subseteq \mathbb{R}^n$ be closed convex sets such that $C_1 \cap ... \cap C_p \neq \varnothing$. Then for any $\mathbf{x}^0 \in \mathbb{R}^n$, the sequence evolved by the iteration $\mathbf{x}^{k+1} = P_{C_p} \circ ... \circ P_{C_1}\mathbf{x}^k$ converges to a point of $C_1 \cap ... \cap C_p$.

Proof. In view of Theorem 3.10 and Proposition 3.11, we need only show that a point $\mathbf{x}$ is a fixed point of $P_{C_1} \circ ... \circ P_{C_p}$ if and only if $\mathbf{x} \in C_1 \cap ... \cap C_p$. The "if" part of this claim is plainly true, so we prove the "only if" part. Let $\mathbf{x}^0$ be a fixed point of $P_{C_p} \circ ... \circ P_{C_1}$. For $q = 1,...,p$, let $\mathbf{x}^q = P_{C_q} \circ ... \circ P_{C_1}\mathbf{x}^0$, and let $\mathbf{x}^*$ be any element of $C_1 \cap ... \cap C_p$, which is assumed to be nonempty. Since $\mathbf{x}$ is a fixed point of $P_{C_p} \circ ... \circ P_{C_1}$, $\mathbf{x}^p = \mathbf{x}^0$. By the firm nonexpansiveness of the the projection operators,

$$\|\mathbf{x}^{q+1} - \mathbf{x}^*\|^2 \leq \|\mathbf{x}^q - \mathbf{x}^*\|^2 - \|\mathbf{x}^{q+1} - \mathbf{x}^q\|^2 \qquad q = 0,...,p-1 \ .$$

Adding and using that $\mathbf{x}^p = \mathbf{x}^0$,

$$\|\mathbf{x}^0 - \mathbf{x}^*\|^2 \;=\; \|\mathbf{x}^p - \mathbf{x}^*\|^2 \;\leq\; \|\mathbf{x}^0 - \mathbf{x}^*\|^2 \;-\; \sum_{q=0}^{p-1} \|\mathbf{x}^{q+1} - \mathbf{x}^q\|^2 \quad ,$$

and therefore $\mathbf{x}^0 = \mathbf{x}^1 = \ldots = \mathbf{x}^{p-1} = \mathbf{x}^p$, and $\mathbf{x}^0 \in C_1 \cap \ldots \cap C_p$. ■

This last result duplicates those of Bregman (1965), Eremin (1965, 1966), Germanov and Spiridonov (1966), Gubin, Polyak, and Raik (1967), and Polyak (1969). In the special case that $C_1,\ldots,C_p$ are closed half-spaces, the result dates back to Von Neuman (1950), Kacmarz (1937), Agmon (1954), and Motzkin and Schoenberg (1954).

(This space intentionally left blank.)

3.4 The Splitting Principle and Categories of Operator Splitting Methods

We now return to our main line of reasoning. We have seen two closely related algorithms: an "explicit" method, forward step, which is relatively easy to implement, and a corresponding "implicit" method, the proximal point algorithm, which has more desirable convergence properties. The difficulty with the proximal point algorithm involves having to invert the operator $(I + \lambda_k T)$.

In numerical linear algebra, particularly as applied to the solution of discretized differential equations (*e.g.* Marchuk 1975, Ferziger 1981), this kind of trade-off between implicit and explicit steps is very common, although the problems typically involve high-dimensional linear, single-valued operators, as opposed to monotone set-valued ones. The idea of operator *splitting* is a popular tool of numerical linear algebra that tries to gain some of the advantages of implicit methods without all of the commensurate computational costs. This idea is typified by the so-called alternating direction implicit (ADI) family of methods. We now describe the adaptation of the notion of splitting to monotone, nonlinear, multivalued operators, an idea that first appeared explicitly in Lions and Mercier (1979).

The fundamental idea of operator splitting is as follows: we wish to find a zero of a maximal monotone operator T, but find inverting operators of the form $(I + \lambda T)$ too difficult to permit the use of the proximal point algorithm. Suppose, however, that there are two maximal monotone operators A and B such that $T = A+B$, where $(I + \lambda A)$ and $(I + \lambda B)$ are relatively easy to invert. Then, by combining forward (explicit) and backward (implicit) steps on A and B alone, all of which should be easy to perform, we might in some sense be able to approximate the effect of a backward step on $T = A+B$. In general,

we define an *operator splitting method* for a maximal monotone operator T to be one that attempts to solve the inclusion $0 \in Tx$ by repeated application of operators of the form I, A, B, $(I + \lambda A)^{-1}$, and $(I + \lambda B)^{-1}$, where A and B are maximal monotone such that $T = A+B$, and λ is some (possibly varying) positive real number.

Many varieties of splitting methods are possible; this thesis will concentrate on four families of splitting schemes: *double-backward, forward-backward, Peaceman-Rachford,* and *Douglas-Rachford.* We will give double-backward the least attention, as it appears to be generally the least useful, while the main emphasis in later chapters will be on Douglas-Rachford methods.

We now proceed to describe these four schemes, and cite general convergence results for them. In the following section, we will develop optimization algorithms that are special cases of these schemes.

3.4.1 The Double-Backward Method

Keeping in mind the motivation that we are trying to obtain some of the benefits of a backward step method on $T = A+B$ without the full burden of evaluating $(I + \lambda T)^{-1} = (I + \lambda(A+B))^{-1}$, perhaps the most natural operator splitting method is given by the *double backward* scheme, as follows:

Definition 3.14. Given a real Hilbert space $\mathcal{H}$, maximal monotone operators T, A, and B such that $T = A+B$, and a sequence $\{\lambda_k\}_{k=0}^{\infty}$ of positive scalars, a sequence $\{x^k\}_{k=0}^{\infty} \subseteq \mathcal{H}$ is said to be generated by the *double-backward scheme* if

$$x^{k+1} = J_{\lambda_k B} J_{\lambda_k A} x^k \quad = (I + \lambda_k B)^{-1}(I + \lambda_k A)^{-1} x^k \qquad \forall\ k \geq 0.$$

In the general case, this approach has been considered Passty (1979), and is depicted in Figure 6. Lions (1978) has also examined this scheme in the special case $B = N_C$, where $C \subseteq \mathcal{H}$ is a nonempty closed convex set. Since Lions' conclusions are no stronger than Passty's, and less general, we give only Passty's main result.

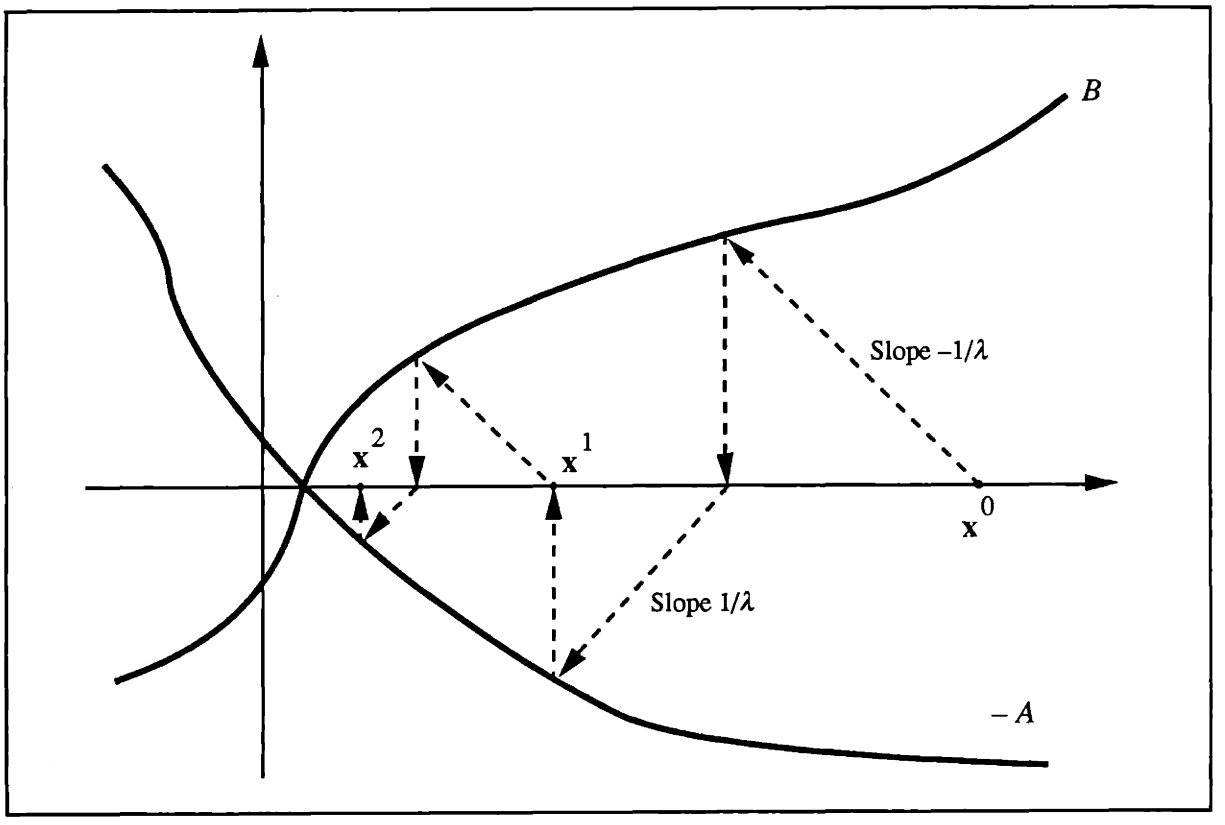


Figure 6. The double-backward iteration. A zero of $A+B$ is a point where the graphs of $-A$ and B intersect.

Theorem 3.11 (Passty 1979). Let $\mathcal{H}$ be any real Hilbert space and let T, A, and B be maximal monotone operators such that $T = A+B$. Further, let $\{x^k\}_{k=0}^{\infty} \subseteq \mathcal{H}$, $\{z^k\}_{k=0}^{\infty} \subseteq \mathcal{H}$, and $\{\lambda_k\}_{k=0}^{\infty} \subseteq (0, \infty)$ be sequences satisfying the conditions

$$x^{k+1} = (I + \lambda_k B)^{-1}(I + \lambda_k A)^{-1}x^k \qquad \forall\, k \geq 0$$

$$z^k = \frac{\left(\sum_{j=0}^{k} \lambda_j x^{j+1}\right)}{\sum_{j=0}^{k} \lambda_j} \qquad \forall\, k \geq 0$$

$$\sum_{k=0}^{\infty} \lambda_k = \infty$$

$$\sum_{k=0}^{\infty} \lambda_k^2 < \infty \quad .$$

Then if zer $T = \varnothing$, $\|z^k\| \to \infty$, while if zer $T \neq \varnothing$, $\{z^k\}$ converges weakly to some zero of T.

This type of convergence is called *ergodic convergence* or *convergence in the mean.* It is not as useful as convergence of $\{x^k\}$, due to the numerical difficulties of computing z^k for large k.

We will not prove Passty's theorem, but will instead attempt to give some insight into why stronger results do not exist. Since $J_{\lambda_k B}$ and $J_{\lambda_k A}$ are firmly nonexpansive, we know from Theorem 3.10 that if we hold λ_k constant at some value $\lambda > 0$, that the sequence $\{x^k\}$ evolved by $x^{k+1} = J_{\lambda B} J_{\lambda A} x^k$ will converge to a fixed point of $J_{\lambda B} J_{\lambda A}$, if one exists. Unfortunately, the fixed points of $J_{\lambda B} J_{\lambda A}$ are generally not the same as the zeroes of $A+B$. To clarify, one has

$$x = J_{\lambda B} J_{\lambda A} x \;\Leftrightarrow\; x \in (I + \lambda A)(I + \lambda B)x$$

$$\Leftrightarrow\; 0 \in y + A(x + \lambda y) \text{ for some } y \in Bx \quad ,$$

whereas

$$0 \in (A+B)x \;\Leftrightarrow\; 0 \in y + A(x) \text{ for some } y \in Bx \quad .$$

A comparison between these two conditions suggests why Passty's result requires (among other things) that $\lambda_k \to 0$, for then the fixed point condition $x = J_{\lambda_k B} J_{\lambda_k A} x$ "approaches", in some sense, the desired condition $0 \in (A+B)x$.

There is one special case in which $0 \in (A+B)x$ and $x = J_{\lambda B} J_{\lambda A} x$ are indeed equivalent:

Proposition 3.12. If $A = N_C$ and $B = N_D$, where C and D are intersecting closed convex sets in a Hilbert space $\mathcal{H}$, then $x = J_{\lambda B} J_{\lambda A} x$ if and only if $0 \in (A+B)x$.

Proof. By Proposition 3.11(i) and Corollary 3.11.2, $J_{\lambda B} J_{\lambda A} = P_D P_C$. Following the proof of Corollary 3.11.3, $x = P_D P_C(x)$ if and only if $x \in C \cap D$, which in turn is true, by the definition of the normal cone operator, if and only if $0 \in (N_C + N_D)x$. ■

This observation accords with the convergence (for any sequence of stepsizes $\{\lambda_k\}_{k=0}^{\infty}$) of the double-backward iteration in the case $A = N_C$ and $B = N_D$, as implied by Corollaries 3.11.2 and 3.11.3.

3.4.2 The Forward-Backward Splitting Scheme

We now turn to the *forward-backward* method, which has much broader applicability than the double-backward method.

Definition 3.15. Let $\mathcal{H}$ be a real Hilbert space, and let A, B: $\mathcal{H} \rightrightarrows \mathcal{H}$ be maximal monotone operators such that $T = A+B$ is maximal. For a sequence $\{\lambda_k\}_{k=0}^{\infty}$ of positive scalars, a sequence $\{x^k\}_{k=0}^{\infty} \subseteq \mathcal{H}$ is said to be generated by the *forward-backward scheme* if

$$x^{k+1} \in J_{\lambda_k B}(I - \lambda_k A)x^k \quad = (I + \lambda_k B)^{-1}(x^k - \lambda_k A x^k) \qquad \forall\, k \geq 0.$$

The forward-backward scheme is depicted in Figure 7. Since it includes a forward step, it inherits certain disadvantages: first, if A is multiple-valued, there may be many choices of x^{k+1} for a given x^k. Second, the method may "jam" if it reaches an iterate x^k for which $Ax^k = \varnothing$.

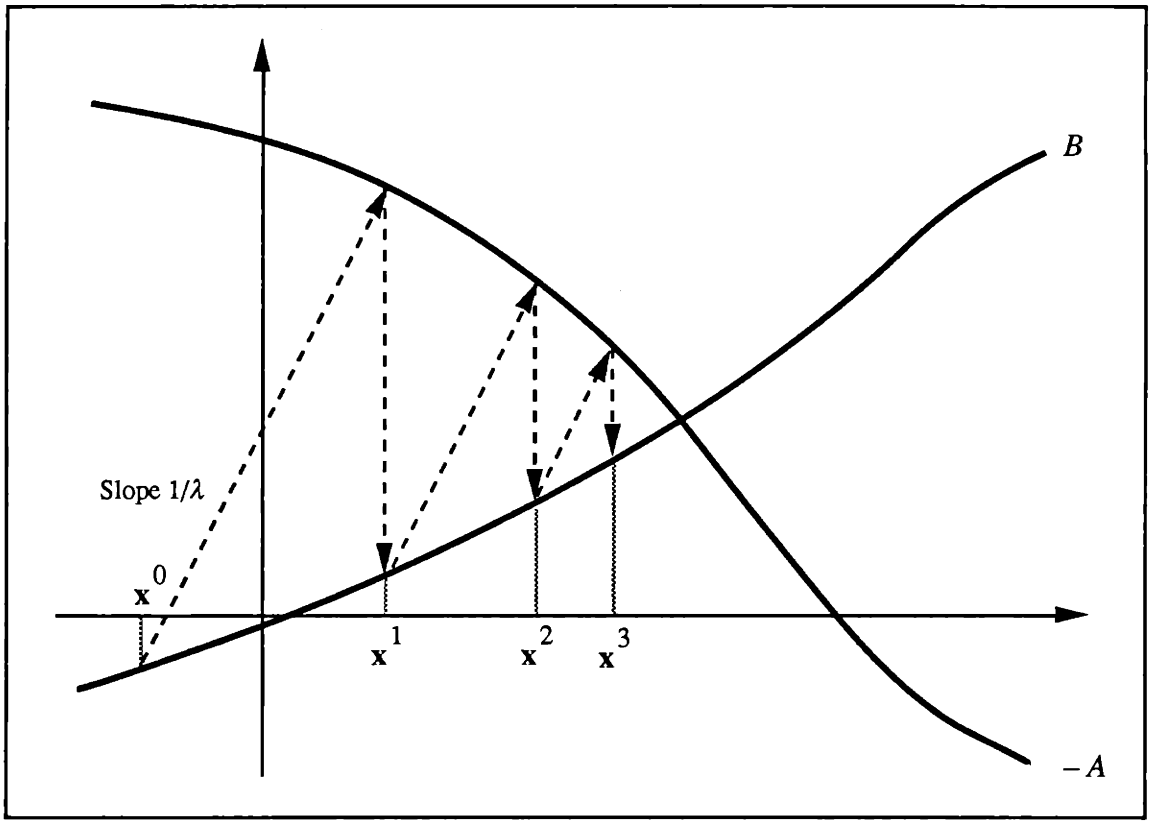


Figure 7. The forward-backward iteration.

However, in comparison with the double-backward method, the forward-backward scheme has the distinct advantage that if $x^{k+1} = x^k$, then x^k is indeed a zero of $A+B$. We have already seen that this kind of property fails to hold for the double-backward scheme. We now make these observations more precise.

Proposition 3.13. Given $\lambda > 0$ and two maximal monotone operators A and B on a Hilbert space, $x \in (I + \lambda B)^{-1}(I - \lambda A)x$ if and only if $0 \in (A+B)x$. If $\{x^k\}$ conforms to the forward backward recursion and $x^j = x^{j+1}$, then $0 \in (A+B)x^j$.

Proof. For the first statement:

$$\begin{aligned} x \in (I + \lambda B)^{-1}(I - \lambda A)x \quad &\Leftrightarrow \quad (I + \lambda B)x \cap (I - \lambda A)x \neq \varnothing \\ &\Leftrightarrow \quad Bx \cap (-Ax) \neq \varnothing \\ &\Leftrightarrow \quad 0 \in (A+B)x \ . \end{aligned}$$

The second claim follows immediately. ■

Unfortunately, if A is multiple-valued, the converse of the second assertion above may not hold, that is, $0 \in (A+B)x^j$ does *not* imply $x^j = x^{j+1}$. This is because, even if $0 \in (A+B)x^j$, there may be members y of $-Ax^j$ that are not members of Bx^j. If the forward step on A is taken in the direction of such a y, then we will have $x^j \neq x^{j+1}$.

If one considers the case in which B is the maximal monotone operator $N_{\mathcal{H}} = \{ (x, 0) \mid x \in \mathcal{H} \}$, the method reduces to the forward step iteration $x^{k+1} \in (I - \lambda_k A)x^k$. In view of the special conditions needed to guarantee convergence of the forward step method, it is clear that the forward-backward method will not converge in general.

Many special cases of the forward-backward method are known to converge. The best known is the *gradient projection* method, due to Goldstein (1964) and Levitin and Polyak (1966). In this case, B is the normal cone operator N_C of some nonempty closed convex set $C \subseteq \mathcal{H}$, and A is the (single-valued) gradient map ∇f associated with a convex

function that is differentiable throughout C. To obtain convergence, one must also assume that ∇f obeys a Lipschitz condition:

Theorem 3.12. Let $B = N_C$, where C is some nonempty, closed, convex subset of a Hilbert space $\mathcal{H}$. Let f: $\mathcal{H} \rightarrow \mathbb{R}$ be a convex function, differentiable on C, such that ∇f meets the Lipschitz condition

$$\|\nabla f(x) - \nabla f(x')\| \leq L\|x - x'\| \quad \forall\, x, x' \in C$$

for some $L \geq 0$. Then, for $A = \partial f$, and any sequence of stepsizes $\{\lambda_k\}_{k=0}^{\infty}$ such that

$$0 < \inf_{k \geq 0} \lambda_k \leq \sup_{k \geq 0} \lambda_k < \frac{2}{L} \quad ,$$

the forward-backward method

$$x^{k+1} \in (I + \lambda_k B)^{-1}(x^k - \lambda_k A x^k) = P_C(x^k - \lambda_k \nabla f(x^k)) \qquad \forall\, k \geq 0$$

converges to the unique zero of $A+B$, that is, the unique minimizer of f on C.

The proof of this theorem follows directly from Theorem 5.1 of Levitin and Polyak (1966), which contains somewhat sharper results. One should note the resemblance of the conditions on A to those of Theorem 3.3. Bertsekas (1976) gives further results for the gradient projection method, using line search in place of predetermined stepsizes.

Another special case of the forward-backward scheme is the *subgradient projection method*, in which $B = N_C$ and $A = \partial f$, where f: $\mathcal{H} \rightarrow \mathbb{R}$ is a convex function, defined throughout C but not necessarily differentiable. Results for this method may be found in Polyak (1987), and use rules for determining stepsizes that resemble those for the unconstrained subgradient method. In one stepsize rule, the $\{\lambda_k\}$ must obey

$$\lim_{k \to \infty} \lambda_k = 0 \qquad \sum_{k=0}^{\infty} \lambda_k = \infty \ ,$$

and ∂f must obey an additional boundedness condition; this rule resembles unconstrained subgradient stepsize rules due to Ermol'ev (1966) and Polyak (1967). Another stepsize rule resembles the unconstrained subgradient rule of Eremin (1966) and Polyak (1969), and requires knowing the optimal objective value.

For completely general maximal monotone operators A and B, the forward-backward method appears to have first been formulated by Passty (1979). However the only known result in this setting is rather weak:

Theorem 3.13 (Passty 1979). Let $\mathcal{H}$ be any real Hilbert space and let T, A, and B be maximal monotone operators such that $T = A+B$. Further, let $\{x^k\}_{k=0}^{\infty}$, $\{y^k\}_{k=0}^{\infty}$, $\{z^k\}_{k=0}^{\infty} \subseteq \mathcal{H}$ and $\{\lambda_k\}_{k=0}^{\infty} \subseteq (0, \infty)$ be sequences satisfying the conditions, for all $k \geq 0$,

$$x^{k+1} = (I + \lambda_k B)^{-1}(x^k - \lambda_k y^k)$$

$$y^k \in Ax^k$$

$$z^k = \frac{\left(\sum_{j=0}^{k} \lambda_j x^{j+1}\right)}{\sum_{j=0}^{k} \lambda_j} \ ,$$

where $\{y^k\}$ is bounded, and

$$\sum_{k=0}^{\infty} \lambda_k = \infty \qquad \sum_{k=0}^{\infty} \lambda_k^2 < \infty \ .$$

Then if zer $T = \varnothing$, $\|z^k\| \to \infty$, while if zer $T \neq \varnothing$, $\{z^k\}$ converges weakly to some zero of T.

This theorem has all the disadvantages of Theorem 3.11's results for the double-backward method, with the additional drawback that it may be very difficult in practice to ensure that $\{y^k\}$ is indeed a bounded sequence.

If one requires that A^{-1} be strongly monotone, then the situation becomes much more favorable, as discovered by Gabay (1983). Tseng (1988) has recently refined Gabay's result to permit varying stepsizes. We will now give Tseng's result and a version of his proof:

Theorem 3.14 (Tseng 1988). Let A and B be maximal monotone operators on $\mathbb{R}^n$ such that $\text{zer}(A+B) \neq \varnothing$ and A^{-1} is strongly monotone with modulus α. Let $\{\lambda_k\}_{k=0}^{\infty} \subseteq (0, \infty)$ and $\{\mathbf{x}^k\}_{k=0}^{\infty} \subseteq \mathbb{R}^n$ be sequences such that

$$\mathbf{x}^{k+1} = (I + \lambda_k B)^{-1}(\mathbf{x}^k - \lambda_k A\mathbf{x}^k) \qquad \forall\, k \geq 0$$

$$0 < \inf_{k \geq 0} \lambda_k \leq \sup_{k \geq 0} \lambda_k < 2\alpha$$

(note that A must be single-valued). Then $\{\mathbf{x}^k\}$ converges to a zero of $A+B$.

Proof. Let $\mathbf{x}^*$ be any member of $\text{zer}(A+B)$, and let $\mathbf{y}^* = A(\mathbf{x}^*)$. Then $-\mathbf{y}^* \in B\mathbf{x}^*$. For all $k \geq 0$, define

$$\mathbf{y}^k = A(\mathbf{x}^k) \qquad\qquad \mathbf{z}^k = \frac{1}{\lambda_k}(\mathbf{x}^k - \mathbf{x}^{k+1}) - \mathbf{y}^k \quad .$$

Note that

$$\begin{aligned} & \mathbf{x}^{k+1} = (I + \lambda_k B)^{-1}(\mathbf{x}^k - \lambda_k \mathbf{y}^k) \\ \Rightarrow \quad & (I + \lambda_k B)\mathbf{x}^{k+1} \ni \mathbf{x}^k - \lambda_k \mathbf{y}^k \\ \Leftrightarrow \quad & \lambda_k B\mathbf{x}^{k+1} \ni \mathbf{x}^k - \mathbf{x}^{k+1} - \lambda_k \mathbf{y}^k \\ \Leftrightarrow \quad & \mathbf{z}^k \in B\mathbf{x}^{k+1} \quad . \end{aligned}$$

Also, by rearranging the definition of $\mathbf{z}^k$, one has

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \lambda_k(\mathbf{y}^k + \mathbf{z}^k) \quad .$$

Using this identity, we have for all $k \geq 0$ that

$$\begin{aligned}
& \langle \mathbf{x}^{k+1} - \mathbf{x}^*, \mathbf{y}^k - \mathbf{y}^* \rangle + \lambda_k \langle \mathbf{z}^k + \mathbf{y}^*, \mathbf{y}^k - \mathbf{y}^* \rangle \\
= \; & \langle \mathbf{x}^k - \mathbf{x}^*, \mathbf{y}^k - \mathbf{y}^* \rangle - \lambda_k \langle \mathbf{y}^k + \mathbf{z}^k, \mathbf{y}^k - \mathbf{y}^* \rangle + \lambda_k \langle \mathbf{z}^k + \mathbf{y}^*, \mathbf{y}^k - \mathbf{y}^* \rangle \\
= \; & \langle \mathbf{x}^k - \mathbf{x}^*, \mathbf{y}^k - \mathbf{y}^* \rangle - \lambda_k \langle \mathbf{y}^k - \mathbf{y}^*, \mathbf{y}^k - \mathbf{y}^* \rangle \\
\geq \; & \alpha \| \mathbf{y}^k - \mathbf{y}^* \|^2 - \lambda_k \| \mathbf{y}^k - \mathbf{y}^* \|^2 \\
= \; & (\alpha - \lambda_k) \| \mathbf{y}^k - \mathbf{y}^* \|^2 \quad ,
\end{aligned}$$

where the inequality follows from the strong monotonicity of A^{-1}. Since $\mathbf{z}^k \in B\mathbf{x}^{k+1}$ and $-\mathbf{y}^* \in B\mathbf{x}^*$,

$$\langle \mathbf{x}^{k+1} - \mathbf{x}^*, \mathbf{z}^k + \mathbf{y}^* \rangle \geq 0 \quad ,$$

so, adding the previous inequality, one obtains

$$\begin{aligned}
& \langle \mathbf{x}^{k+1} - \mathbf{x}^*, \mathbf{y}^k + \mathbf{z}^k \rangle + \lambda_k \langle \mathbf{z}^k + \mathbf{y}^*, \mathbf{y}^k - \mathbf{y}^* \rangle \geq (\alpha - \lambda_k) \| \mathbf{y}^k - \mathbf{y}^* \|^2 \\
\Leftrightarrow \quad & \langle \mathbf{x}^{k+1} - \mathbf{x}^*, \tfrac{1}{\lambda_k}(\mathbf{x}^k - \mathbf{x}^{k+1}) \rangle + \lambda_k \langle \mathbf{z}^k + \mathbf{y}^*, \mathbf{y}^k - \mathbf{y}^* \rangle \geq (\alpha - \lambda_k) \| \mathbf{y}^k - \mathbf{y}^* \|^2 \\
\Leftrightarrow \quad & \tfrac{1}{\lambda_k} \langle \mathbf{x}^{k+1} - \mathbf{x}^*, \mathbf{x}^k - \mathbf{x}^{k+1} \rangle \geq (\alpha - \lambda_k) \| \mathbf{y}^k - \mathbf{y}^* \|^2 - \lambda_k \langle \mathbf{z}^k + \mathbf{y}^*, \mathbf{y}^k - \mathbf{y}^* \rangle \quad .
\end{aligned}$$

From the identity

$$\begin{aligned}
\| \mathbf{x}^k - \mathbf{x}^* \|^2 &= \| \mathbf{x}^k - \mathbf{x}^{k+1} + \mathbf{x}^{k+1} - \mathbf{x}^* \|^2 \\
&= \| \mathbf{x}^k - \mathbf{x}^{k+1} \|^2 + 2 \langle \mathbf{x}^{k+1} - \mathbf{x}^*, \mathbf{x}^k - \mathbf{x}^{k+1} \rangle + \| \mathbf{x}^{k+1} - \mathbf{x}^* \|^2
\end{aligned}$$

one obtains

$$\langle \mathbf{x}^{k+1} - \mathbf{x}^*, \mathbf{x}^k - \mathbf{x}^{k+1} \rangle = \tfrac{1}{2} [\| \mathbf{x}^k - \mathbf{x}^* \|^2 - \| \mathbf{x}^{k+1} - \mathbf{x}^* \|^2 - \| \mathbf{x}^k - \mathbf{x}^{k+1} \|^2] \quad .$$

Therefore,

$$\frac{1}{2\lambda_k}[\|\mathbf{x}^k - \mathbf{x}^*\|^2 - \|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 - \|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2]$$
$$\geq (\alpha - \lambda_k)\|\mathbf{y}^k - \mathbf{y}^*\|^2 - \lambda_k\langle \mathbf{z}^k + \mathbf{y}^*, \mathbf{y}^k - \mathbf{y}^*\rangle$$

Rearranging,

$$\begin{aligned}\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \leq\ & \|\mathbf{x}^k - \mathbf{x}^*\|^2 - \|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2 \\ & + 2\lambda_k{}^2\langle \mathbf{z}^k + \mathbf{y}^*, \mathbf{y}^k - \mathbf{y}^*\rangle \\ & + (2\lambda_k{}^2 - 2\lambda_k\alpha)\|\mathbf{y}^k - \mathbf{y}^*\|^2 \quad .\end{aligned}$$

Now,

$$\begin{aligned}\|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2 &= \|\lambda_k(\mathbf{y}^k + \mathbf{z}^k)\|^2 \\ &= \lambda_k{}^2\|\mathbf{y}^k - \mathbf{y}^* + \mathbf{z}^k + \mathbf{y}^*\|^2 \\ &= \lambda_k{}^2\|\mathbf{y}^k - \mathbf{y}^*\|^2 + 2\lambda_k{}^2\langle \mathbf{z}^k + \mathbf{y}^*, \mathbf{y}^k - \mathbf{y}^*\rangle + \lambda_k{}^2\|\mathbf{z}^k + \mathbf{y}^*\|^2 \quad ,\end{aligned}$$

so, substituting for $\|\mathbf{x}^k - \mathbf{x}^{k+1}\|^2$ gives

$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \leq \|\mathbf{x}^k - \mathbf{x}^*\|^2 - \lambda_k{}^2\|\mathbf{z}^k + \mathbf{y}^*\|^2 - \lambda_k(2\alpha - \lambda_k)\|\mathbf{y}^k - \mathbf{y}^*\|^2 \quad .$$

Let

$$\varepsilon = \min\left\{\inf_{k\geq 0}\{\lambda_k\}, 2\alpha - \sup_{k\geq 0}\{\lambda_k\}\right\} > 0 \quad .$$

Then, for all $k \geq 0$,

$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \leq \|\mathbf{x}^k - \mathbf{x}^*\|^2 - \varepsilon^2\|\mathbf{z}^k + \mathbf{y}^*\|^2 - \varepsilon^2\|\mathbf{y}^k - \mathbf{y}^*\|^2 \quad , \qquad (\dagger)$$

hence $\|\mathbf{x}^{k+1} - \mathbf{x}^*\|^2 \leq \|\mathbf{x}^k - \mathbf{x}^*\|^2$ for all $k \geq 0$, and $\{\mathbf{x}^k\}$ is bounded. Adding the inequality (†) over $k = 0,\ldots,K-1$, we have

$$\|\mathbf{x}^K - \mathbf{x}^*\|^2 \leq \|\mathbf{x}^0 - \mathbf{x}^*\|^2 - \varepsilon^2\left[\sum_{k=0}^{K-1}\|\mathbf{z}^k - \mathbf{y}^*\|^2 + \sum_{k=0}^{K-1}\|\mathbf{y}^k - \mathbf{y}^*\|^2\right] \quad ,$$

and, letting $K \to \infty$, one concludes that

$$\lim_{k \to \infty} \mathbf{z}^k = -\mathbf{y}^* \qquad \lim_{k \to \infty} \mathbf{y}^k = \mathbf{y}^* \quad .$$

Since $\{\mathbf{x}^k\}$ is bounded, it possesses a limit point $\mathbf{x}^\infty$. Let $\{\mathbf{x}^{k(j)}\}_{j=0}^{\infty}$ be subsequence of $\{\mathbf{x}^k\}$ converging to $\mathbf{x}^\infty$. Then

$$\begin{aligned}
&\lim_{j \to \infty} (\mathbf{x}^{k(j)}, \mathbf{y}^{k(j)}) = (\mathbf{x}^\infty, \mathbf{y}^*) \\
&(\mathbf{x}^{k(j)}, \mathbf{y}^{k(j)}) \in A \quad \forall\, j \geq 0 \\
&\lim_{j \to \infty} (\mathbf{x}^{k(j)}, \mathbf{z}^{k(j)-1}) = (\mathbf{x}^\infty, -\mathbf{y}^*) \\
&(\mathbf{x}^{k(j)}, \mathbf{z}^{k(j)-1}) \in B \quad \forall\, j \geq 0 \quad .
\end{aligned}$$

Therefore, by the maximal monotonicity of A and B , Proposition 3.2 gives that $(\mathbf{x}^\infty, \mathbf{y}^*) \in A$ and $(\mathbf{x}^\infty, -\mathbf{y}^*) \in B$, hence $\mathbf{x}^\infty \in \operatorname{zer}(A+B)$. Since $\mathbf{x}^* \in \operatorname{zer}(A+B)$ was chosen arbitrarily, we can substitute $\mathbf{x}^\infty$ for $\mathbf{x}^*$, obtaining $\|\mathbf{x}^{k+1} - \mathbf{x}^\infty\|^2 \leq \|\mathbf{x}^k - \mathbf{x}^\infty\|^2$ for all $k \geq 0$. It follows that $\mathbf{x}^*$ is the only limit point of $\{\mathbf{x}^k\}$, and $\{\mathbf{x}^k\}$ converges to $\mathbf{x}^*$. ■

Note the resemblance of Theorem 3.14 to Theorem 3.3. Indeed, if one disregards the possibility of approximate computation, Theorem 3.3 represents the special case of Theorem 3.14 in which B is the *zero operator* $0 = N_{\mathbb{R}^n} = \{(\mathbf{x}, \mathbf{0}) \mid \mathbf{x} \in \mathbb{R}^n\}$.

3.4.3 Peaceman-Rachford Methods

In numerical analysis, one common variant of the forward-backward splitting scheme is obtained by exchanging the roles of the operators A and B at each iteration, yielding the recursion

$$\mathbf{x}^{k+1} \in (I + \lambda_k B)^{-1}(I - \lambda_k A)(I + \lambda_k A)^{-1}(I - \lambda_k B)\mathbf{x}^k \quad . \tag{PR}$$

For special, linear choices of A and B, related to the approximate solution of the heat equation on a planar grid, a method of this type was first proposed by Peaceman and

Rachford (1955). Kellogg (1969) extended the analysis of this approach to the case where A and B are monotone and possibly nonlinear, but still single-valued, operators. Finally, Lions and Mercier (1979) considered the case in which A and B are general multivalued monotone operators.

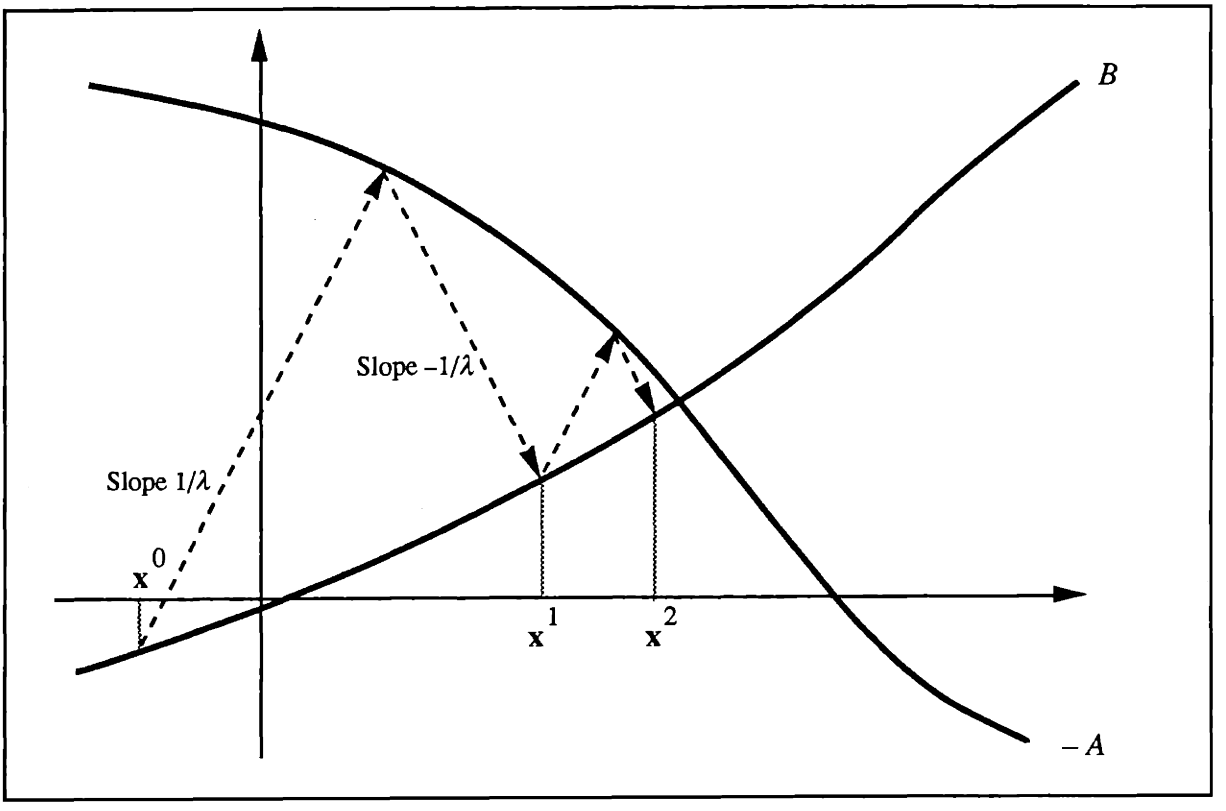


Figure 8. The Peaceman-Rachford iteration.

Definition 3.16. Let $\mathcal{H}$ be a real Hilbert space, and let A, B: $\mathcal{H} \rightrightarrows \mathcal{H}$ be maximal monotone operators such that $T = A+B$ is maximal. For a sequence $\{\lambda_k\}_{k=0}^{\infty}$ of positive scalars, a sequence $\{x^k\}_{k=0}^{\infty} \subseteq \mathcal{H}$ is said to be generated by the *loose Peaceman-Rachford recursion* if it obeys

$$\mathbf{x}^{k+1} \in (I + \lambda_k B)^{-1}(I - \lambda_k A)(I + \lambda_k A)^{-1}(I - \lambda_k B)\mathbf{x}^k \qquad \forall\ k \geq 0.$$

This method is depicted in Figure 8. Lions and Mercier also introduced a special case of the method in which the choice of the next iterate is always unique, even if neither A nor B is single-valued. We now explain this variation, which we call the *tight*, as opposed to loose, version of the algorithm. For multivalued operators, all known convergence results apply only to the tight version. The loose version is known to converge only for single-valued operators, in which case it is equivalent to the tight version.

The loose Peaceman-Rachford iteration (PR) may be implemented by the following steps:

(i) Choose any $\tilde{b}^k \in Bx^k$

(ii) Find the unique $(y^{k+1}, a^{k+1}) \in A$ such that $y^{k+1} + \lambda_k a^{k+1} = x^k - \lambda_k \tilde{b}^k$

(iii) Choose any $\tilde{a}^{k+1} \in Ay^{k+1}$

(iv) Find the unique $(x^{k+1}, b^{k+1}) \in B$ such that $x^{k+1} + \lambda_k b^{k+1} = y^{k+1} - \lambda_k \tilde{a}^{k+1}$.

The existence and uniqueness in steps (ii) and (iv) is a direct consequence of the properties of the resolvents $J_{\lambda A}$ and $J_{\lambda B}$, as established in Corollary 3.6.3. Notice that the arbitrary choice in step (iii) redundant, as step (ii) produces as a side-effect the vector $a^{k+1} = \frac{1}{\lambda_k}(x^k - \lambda_k b^k - y^k)$, which must lie in Ay^k. Similarly, at every iteration except the first, the arbitrary choice of $\tilde{b}^k$ is unnecessary, because step (iv) of the previous iteration yields the vector

$$b^k = \frac{1}{\lambda_{k-1}}(y^k - \lambda_k \tilde{a}^k - x^k) \quad ,$$

which must be in Bx^k. Combining these two observations, we see that one possible implementation of steps (i)-(iv) is, given some initial $(x^0, b^0) \in B$, to execute repeatedly

(i') Find the unique $(y^{k+1}, a^{k+1}) \in A$ such that $y^{k+1} + \lambda_k a^{k+1} = x^k - \lambda_k b^k$

(ii') Find the unique $(x^{k+1}, b^{k+1}) \in B$ such that $x^{k+1} + \lambda_k b^{k+1} = y^{k+1} - \lambda_k a^{k+1}$.

Note that once $(x^0, b^0) \in B$ is chosen, (i')-(ii') determine a unique sequence $\{(x^k, b^k)\}_{k=0}^{\infty} \subseteq B$, which is the same as would be obtained by always choosing $\tilde{a}^k = a^k$ and $\tilde{b}^k = b^k$ is steps (i)-(iv) above. Thus, we have

Definition 3.17. Given two maximal monotone operators A and B on a Hilbert space $\mathcal{H}$, a sequence $\{(x^k, b^k)\}_{k=0}^{\infty} \subseteq B$ is said to be generated by the *tight* or *canonical Peaceman-Rachford* method if for all $k \geq 0$, (x^{k+1}, b^{k+1}) may be obtained from (x^k, b^k) by the procedure (i')-(ii') above.

The proof of the following result should now be immediate:

Proposition 3.14. (i) For each $(x^0, b^0) \in B$ and positive stepsize sequence $\{\lambda_k\}_{k=0}^{\infty}$, there exists exactly one sequence $\{(x^k, b^k)\}_{k=0}^{\infty} \subseteq B$ conforming to the tight Peaceman-Rachford recursion. (ii) The sequence $\{x^k\}_{k=0}^{\infty} \subseteq \text{dom } B$ defined in this way always conforms to the loose Peaceman-Rachford recursion for A, B, and $\{\lambda_k\}$.

It is not strictly necessary to have $(x^0, b^0) \in B$, as (ii') guarantees that $(x^1, b^1) \in B$, even if $(x^0, b^0) \notin B$.

Figures 9 and 10 illustrate the difference between the "loose" and "tight" Peaceman-Rachford iterations, as we have defined them. In the rest of this section, all calculations should be assumed to be by the "tight" method unless otherwise indicated.

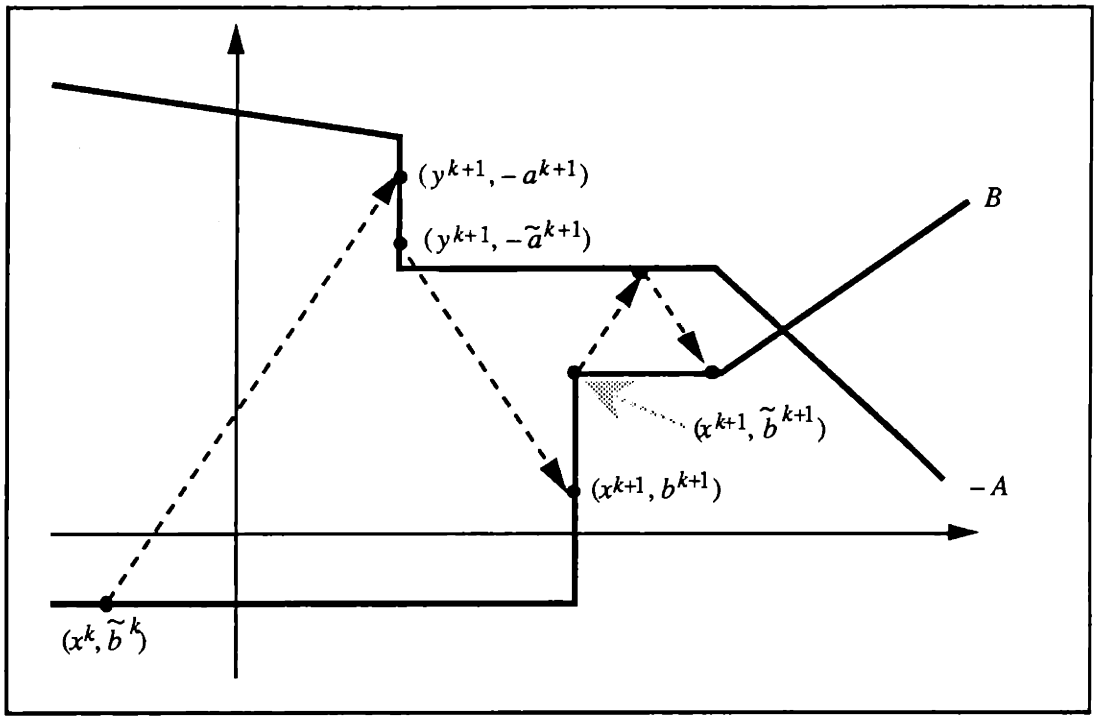


Figure 9. The "loose" Peaceman-Rachford iteration.

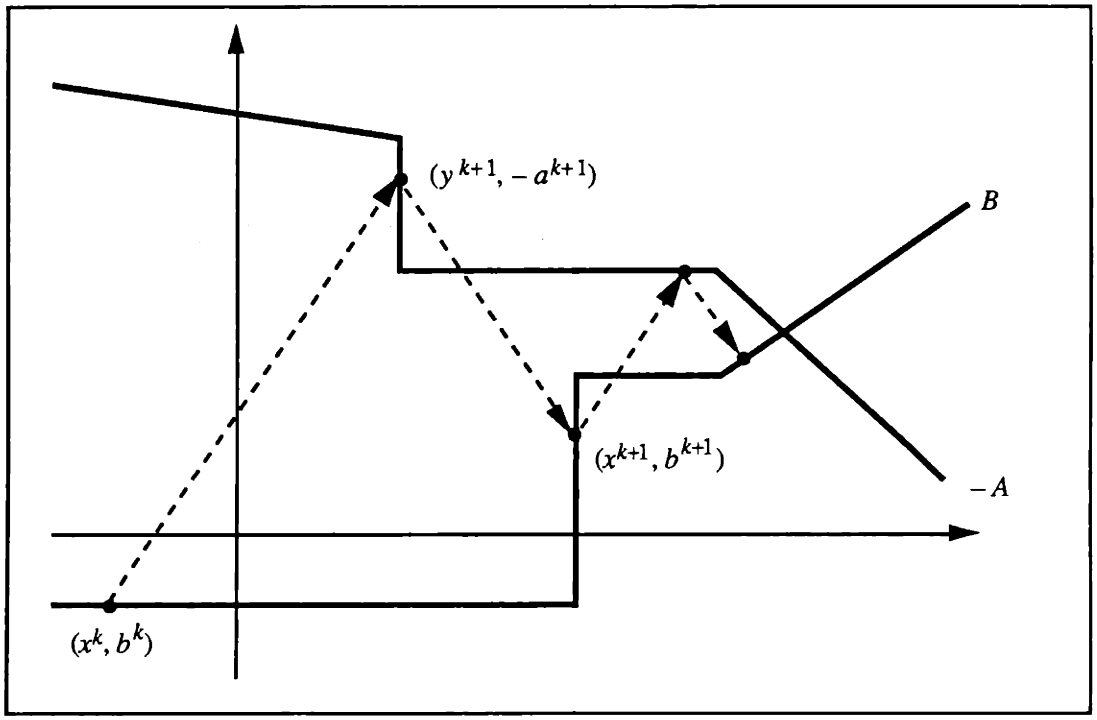


Figure 10. The "tight" Peaceman-Rachford iteration.

We now give an alternate description of the tight version algorithm, again due to Lions and Mercier (1979). To best understand this description (and a similar description of the *Douglas*-Rachford splitting scheme in the next section), it is best to keep in mind the following simple identities: given any element z of a Hilbert space $\mathcal{H}$, $\lambda > 0$, and maximal monotone operator T on $\mathcal{H}$, z has a unique representation as $x + \lambda y$, where $(x, y) \in T$. Then

$$\begin{aligned} J_{\lambda T}(z) &= x \\ (I - J_{\lambda T})(z) &= \lambda y \\ (2J_{\lambda T} - I)(z) &= 2x - (x + \lambda y) = x - \lambda y \quad . \end{aligned}$$

In particular, the operator $2J_{\lambda T} - I$ "flips" the sign of the λy component of $z = x + \lambda y$.

Proposition 3.15. Let A and B be maximal monotone operators and let $\{\lambda_k\}$ be a positive constant sequence, that is, $\lambda_k = \lambda > 0$ for all k. Let $z^0 = x^0 + \lambda b^0$ for any $(x^0, b^0) \in B$, and let $\{z^k\}$ be the sequence given by

$$z^{k+1} = (2J_{\lambda A} - I)(2J_{\lambda B} - I)z^k \qquad \forall\, k \geq 0 \quad .$$

Then the sequence $\{(x^k, b^k)\} \subseteq B$ generated by the tight Peaceman-Rachford scheme starting from (x^0, b^0) has the properties, for all $k \geq 0$,

$$z^k = x^k + \lambda b^k \qquad x^k = J_{\lambda B}\, z^k \qquad b^k = \tfrac{1}{\lambda}(z^k - x^k) \quad .$$

Proof. We show inductively that $z^k = x^k + \lambda b^k$ for all $k \geq 0$. The remaining formulas are then immediate. The base case, $z^0 = x^0 + \lambda b^0$, is true by hypothesis. Now assume that $z^k = x^k + \lambda b^k$ for some k. Then, noting that $(x^k, b^k) \in B$,

$$(2J_{\lambda B} - I)z^k = (2J_{\lambda B} - I)(x^k + \lambda b^k) = 2x^k - (x^k + \lambda b^k) = x^k - \lambda b^k \quad .$$

By the definition of step (i'), one has

$$y^{k+1} + \lambda_k a^{k+1} = x^k - \lambda_k b^k \ ,$$

where $(y^{k+1}, a^{k+1}) \in A$, hence

$$\begin{aligned} z^{k+1} &= (2J_{\lambda A} - I)(2J_{\lambda B} - I)z^k \\ &= (2J_{\lambda A} - I)(x^k - \lambda b^k) \\ &= (2J_{\lambda A} - I)(y^{k+1} + \lambda_k a^{k+1}) \\ &= y^{k+1} - \lambda_k a^{k+1} \\ &= x^{k+1} + \lambda_k b^{k+1} \ , \end{aligned}$$

where the last equality follows from (ii'). So, $z^{k+1} = x^{k+1} + \lambda_k b^{k+1}$, establishing the induction. ■

The tight Peaceman-Rachford iteration's fixed-point properties are an improvement over those of the forward-backward method.:

Proposition 3.16. The operator $(2J_{\lambda A} - I)(2J_{\lambda B} - I)$ is single-valued and its set of fixed is

$$T_\lambda^* = \{ \, x + \lambda b \mid (x, b) \in B, (x, -b) \in A \, \} \ .$$

Proof. Since $J_{\lambda A}$ and $J_{\lambda B}$ are single-valued, so is $(2J_{\lambda A} - I)(2J_{\lambda B} - I)$. Consider any point $z \in \mathcal{H}$. Then z can be written exactly one way as $x + \lambda b$, where $(x, b) \in B$. Then

$$\begin{aligned} & z = (2J_{\lambda A} - I)(2J_{\lambda B} - I)z \\ \Leftrightarrow \quad & x + \lambda b = (2J_{\lambda A} - I)(x - \lambda b) \\ \Leftrightarrow \quad & x + \lambda b = 2J_{\lambda A}(x - \lambda b) - x + \lambda b \\ \Leftrightarrow \quad & x = J_{\lambda A}(x - \lambda b) \\ \Leftrightarrow \quad & -b \in Ax \\ \Leftrightarrow \quad & z \in T_\lambda^* \ . \end{aligned}$$ ■

One should further note that $\mathrm{zer}(A+B) = \{J_{\lambda B}(z) \mid z \in T_\lambda^*\}$, so the zeroes of $A+B$ are easily obtained from the fixed points of $(2J_{\lambda A} - I)(2J_{\lambda B} - I)$ by applying the operator $J_{\lambda B}$.

It remains to be shown that the Peaceman-Rachford method converges. By Proposition 3.8, $(2J_{\lambda A} - I)(2J_{\lambda B} - I)$ is nonexpansive, but it is not necessarily a contraction mapping. Unfortunately, convergence is *not* completely general, as shown in Figure 11.

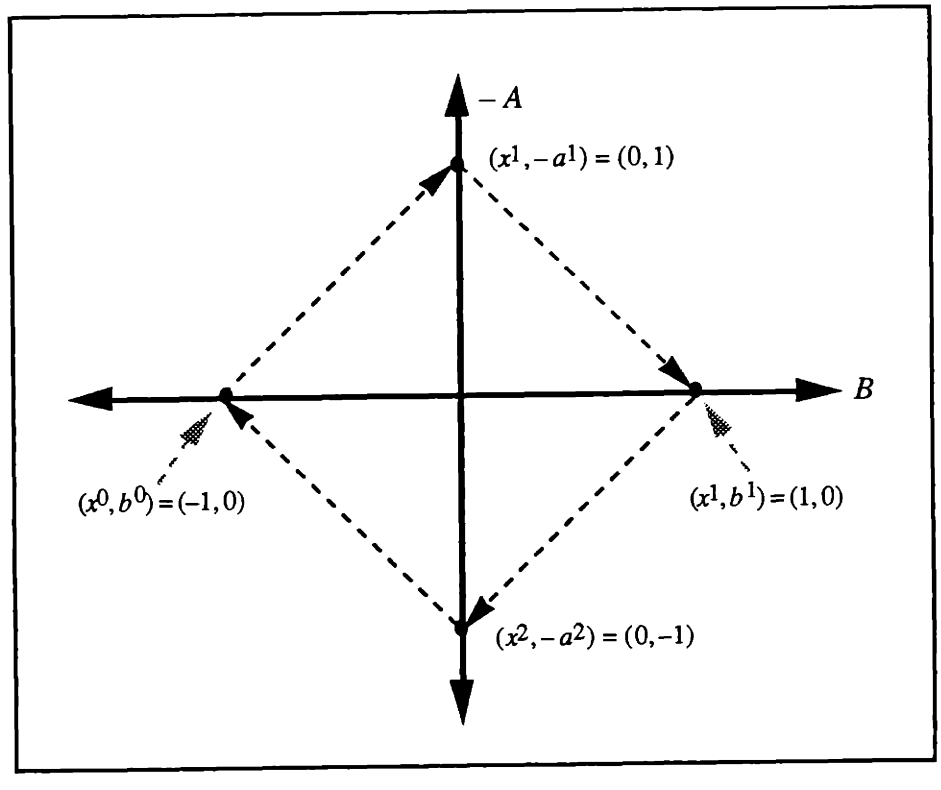


Figure 11. Example of the failure of the Peaceman-Rachford method to converge for two operators in $\mathbb{R}^1$. Here, $A = \{ (0, y) \mid y \in \mathbb{R}\}$ and $B = \{ (x, 0) \mid x \in \mathbb{R}\}$. It may be easily confirmed that $z^k = (-1)^k z^0$, and hence that the method will not converge unless it is started exactly at the unique zero of $A+B$, 0.

We now essentially repeat the main Peaceman-Rachford convergence result of Lions and Mercier (1979), using a simplified notation, and considering only the case of finite dimension. Before the main result, we need a definition:

Definition 3.18. An operator $T: \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is said to have the *finite dimensional Lions-Mercier property* if it is single-valued, and for any $\mathbf{x} \in \text{dom } T$ and convergent sequence $\{x^k\} \subseteq \text{dom } T$ such that $\{T\mathbf{x}^k\}$ is also convergent, one also has

$$\lim_{k \to \infty} \langle T\mathbf{x}^k - T\mathbf{x}, \mathbf{x}^k - \mathbf{x} \rangle = 0 \quad \Longrightarrow \quad \lim_{k \to \infty} \mathbf{x}^k = \mathbf{x} \quad .$$

It is now possible to state a finite-dimensional version Lions and Mercier's main result for Peaceman-Rachford Splitting:

Proposition 3.17. Let A, B: $\mathbb{R}^n \rightrightarrows \mathbb{R}^n$ be maximal monotone operators, at least one of which has the finite-dimensional Lions-Mercier property. Also suppose that $\text{zer}(A+B)$ is nonempty. Then the Peaceman-Rachford method $\mathbf{z}^{k+1} = (2J_{\lambda A} - I)(2J_{\lambda B} - I)\mathbf{z}^k$, with constant stepsize $\lambda > 0$, converges to an element of T_λ^* for any starting point $\mathbf{z}^0 \in \mathbb{R}^n$.

Proof. Let $(\mathbf{x}^0, \mathbf{b}^0)$ be the unique pair in B such that $\mathbf{x}^0 + \lambda\mathbf{b}^0 = \mathbf{z}^0$, and let the sequences $\{\mathbf{x}^k\}$, $\{\mathbf{b}^k\}$, $\{\mathbf{y}^k\}$, and $\{\mathbf{a}^k\}$ be as defined in (i')-(ii'), hence $\mathbf{z}^k = \mathbf{x}^k + \lambda\mathbf{b}^k$ for all k by Proposition 3.15. Let $\mathbf{z}^*$ be any element of T_λ^*, that is $\mathbf{z}^k = \mathbf{x}^* + \lambda\mathbf{b}^*$, where $(\mathbf{x}^*, \mathbf{b}^*) \in B$, and $(\mathbf{x}^*, -\mathbf{b}^*) \in A$. Let $\mathbf{w}^* = \mathbf{x}^* - \lambda\mathbf{b}^*$, and $\mathbf{w}^k = \mathbf{y}^{k+1} + \lambda\mathbf{a}^{k+1} = \mathbf{x}^k - \lambda\mathbf{b}^k$ for all k. Then

$$\mathbf{w}^{k+1} = (2J_{\lambda B} - I)(2J_{\lambda A} - I)\mathbf{w}^k \quad \forall\, k \geq 0$$

$$\mathbf{w}^* = (2J_{\lambda B} - I)(2J_{\lambda A} - I)\mathbf{w}^* \quad .$$

It follows from the nonexpansiveness of $(2J_{\lambda A} - I)$ and $(2J_{\lambda B} - I)$ that $\|\mathbf{z}^k - \mathbf{z}^*\|$ and $\|\mathbf{w}^k - \mathbf{w}^*\|$ are nonincreasing, and hence converge to nonnegative limits d_z and d_w, respectively. Now,

$$\|\mathbf{z}^k - \mathbf{z}^*\|^2 = \|\mathbf{x}^k - \mathbf{x}^*\|^2 + 2\lambda\langle \mathbf{x}^k - \mathbf{x}^*, \mathbf{b}^k - \mathbf{b}^*\rangle + \lambda^2\|\mathbf{b}^k - \mathbf{b}^*\|^2$$

$$\|\mathbf{w}^k - \mathbf{w}^*\|^2 = \|\mathbf{x}^k - \mathbf{x}^*\|^2 - 2\lambda\langle \mathbf{x}^k - \mathbf{x}^*, \mathbf{b}^k - \mathbf{b}^*\rangle + \lambda^2\|\mathbf{b}^k - \mathbf{b}^*\|^2 \quad ,$$

which combine to give, along with the monotonicity of B, that

$$\|\mathbf{z}^k - \mathbf{z}^*\|^2 - \|\mathbf{w}^k - \mathbf{w}^*\|^2 = 4\lambda\langle \mathbf{x}^k - \mathbf{x}^*, \mathbf{b}^k - \mathbf{b}^*\rangle \geq 0 \quad .$$

Similarly, as $\mathbf{z}^{k+1} = \mathbf{y}^{k+1} - \lambda\mathbf{a}^{k+1}$ for all k,

$$\|\mathbf{w}^k - \mathbf{w}^*\|^2 = \|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 + 2\lambda\langle \mathbf{y}^{k+1} - \mathbf{y}^*, \mathbf{a}^{k+1} - \mathbf{a}^*\rangle + \lambda^2\|\mathbf{a}^{k+1} - \mathbf{a}^*\|^2$$

$$\|\mathbf{z}^{k+1} - \mathbf{z}^*\|^2 = \|\mathbf{y}^{k+1} - \mathbf{y}^*\|^2 - 2\lambda\langle \mathbf{y}^{k+1} - \mathbf{y}^*, \mathbf{a}^{k+1} - \mathbf{a}^*\rangle + \lambda^2\|\mathbf{a}^{k+1} - \mathbf{a}^*\|^2 \quad ,$$

whence

$$\|\mathbf{w}^k - \mathbf{w}^*\| - \|\mathbf{z}^{k+1} - \mathbf{z}^*\| = 4\lambda\langle \mathbf{y}^{k+1} - \mathbf{y}^*, \mathbf{a}^{k+1} - \mathbf{a}^*\rangle \geq 0 \quad .$$

Taking limits, we have

$$d_z - d_w = 4\lambda \lim_{k \to \infty} \langle \mathbf{x}^k - \mathbf{x}^*, \mathbf{b}^k - \mathbf{b}^*\rangle \geq 0$$

$$d_w - d_z = 4\lambda \lim_{k \to \infty} \langle \mathbf{y}^{k+1} - \mathbf{y}^*, \mathbf{a}^{k+1} - \mathbf{a}^*\rangle \geq 0 \, ,$$

and so $d_z = d_w$ and

$$\lim_{k \to \infty} \langle \mathbf{x}^k - \mathbf{x}^*, \mathbf{b}^k - \mathbf{b}^*\rangle = \lim_{k \to \infty} \langle \mathbf{y}^{k+1} - \mathbf{y}^*, \mathbf{a}^{k+1} - \mathbf{a}^*\rangle = 0 \, .$$

Note also that $\{\mathbf{z}^k\}$ is bounded, and therefore $\{\mathbf{x}^k\} = \{J_{\lambda B}\,\mathbf{z}^k\}$ is bounded by the nonexpansiveness of $J_{\lambda B}$. Then $\{\mathbf{b}^k\} = \{\frac{1}{\lambda}(\mathbf{z}^k - \mathbf{x}^k)\}$ is also bounded.

Now suppose that it is B that has the finite dimensional Lions-Mercier property. Since $\{\mathbf{z}^k\}$ is bounded, it has at least one limit point $\mathbf{z}^\infty$, so let $\{\mathbf{z}^{k(j)}\}_{j=0}^{\infty}$ be a subsequence converging to $\mathbf{z}^\infty$. By the continuity of $J_{\lambda B}$, $\{\mathbf{x}^{k(j)}\}_{j=0}^{\infty}$ and $\{\mathbf{b}^{k(j)}\}_{j=0}^{\infty}$ are also convergent. As $\lim_{j\to\infty}\langle \mathbf{x}^{k(j)} - \mathbf{x}^*, \mathbf{b}^{k(j)} - \mathbf{b}^*\rangle = 0$, we have by the Lions-Mercier property of B that $\lim_{j\to\infty} \mathbf{x}^{k(j)} = \mathbf{x}^*$. Since B is single-valued, it follows that $\lim_{j\to\infty} \mathbf{b}^{k(j)} = \mathbf{b}^*$, and hence $\mathbf{z}^\infty = \lim_{j\to\infty} \mathbf{z}^{k(j)} = \mathbf{z}^*$. Since $\|\mathbf{z}^k - \mathbf{z}^*\|$ is nonincreasing, $\mathbf{z}^\infty = \mathbf{z}^*$ is the only limit point of $\{\mathbf{z}^k\}$, establishing the theorem.

The proof for the case in which it is A that has the finite-dimensional Lions-Mercier property is similar. ■

Corollary 3.17.1. If A or B is strongly monotone and single-valued, then (in finite dimension) the Peaceman-Rachford $\mathbf{z}^{k+1} = (2J_{\lambda A} - I)(2J_{\lambda B} - I)\mathbf{z}^k$ method is convergent, provided zer$(A+B)$ is nonempty.

Proof. Assume that B is strongly monotone and single-valued. Let $\{(\mathbf{x}^k, \mathbf{b}^k)\}_{k=0}^{\infty} \subseteq B$ and $(\mathbf{x}, \mathbf{b})$ be any point on B. Then for some $\alpha > 0$,

$$\langle \mathbf{x}^k - \mathbf{x}, \mathbf{b}^k - \mathbf{b}\rangle \geq \alpha\|\mathbf{x}^k - \mathbf{x}^*\|^2 ,$$

hence $\langle \mathbf{x}^k - \mathbf{x}, \mathbf{b}^k - \mathbf{b}\rangle \to 0$ implies $\mathbf{x}^k \to \mathbf{x}$. Since B is single-valued, it has the finite-dimensional Lions-Mercier property. A similar analysis holds if A is strongly monotone and single-valued. ■

Note that it is *not* sufficient for A or B to be strongly monotone; single-valuedness is essential. For instance, in the example of Figure 11, $A = \{(0, y) \mid y \in \mathbb{R}^1\}$ is strongly monotone, yet the Peaceman-Rachford method does not converge.

3.4.4 Douglas-Rachford Splitting methods

So far, none of the splitting methods we have considered converge for general maximal monotone choices of A and B. We now describe the *Douglas-Rachford* splitting scheme, which does exhibit general convergence, at least when used with a constant stepsize in finite-dimensional spaces.

This method was originally motivated by a power-series analysis of a discretization of the heat equation (Douglas and Rachford 1954). It is based on the iteration

$$\mathbf{x}^{k+1} \in (I + \lambda_k B)^{-1}[(I + \lambda_k A)^{-1}(I - \lambda_k B) + \lambda_k B]\mathbf{x}^k \ . \qquad \text{(DR)}$$

The first operations of the scheme, $(I + \lambda_k A)^{-1}(I - \lambda_k B)$, are the same as in the forward-backward or Peaceman-Rachford methods. However, the forward step is then "corrected" or "cancelled out", in some sense, by the addition of $\lambda_k B\mathbf{x}^k$, and finally a backward step is taken on B. For single-valued operators, the following interpretation is possible: the method takes the same backward step on A that it would have taken if a forward step on B had been taken first; then, it takes a true backward step on B. The method was first analyzed in the general case by Lions and Mercier (1979), whose development we will now follow.

The most general possible implementation of the iteration (DR) is as follows: given $x^k \in \text{dom}\, B$,

(a) Choose any $\tilde{b}^k \in Bx^k$

(b) Find the unique $(y^{k+1}, a^{k+1}) \in A$ such that $y^{k+1} + \lambda_k a^{k+1} = x^k - \lambda_k \tilde{b}^k$

(c) Choose any $\hat{b}^k \in Bx^k$

(d) Find the unique $(x^{k+1}, b^{k+1}) \in B$ such that $x^{k+1} + \lambda_k b^{k+1} = y^{k+1} + \lambda_k \hat{b}^k$.

The existence and uniqueness claimed in (b) and (d) is a consequence of the Representation Lemma (Corollary 3.6.3). Much as in the case of the Peaceman-Rachford scheme, the arbitrary choices in steps (a) and (c) above can be eliminated. At any iteration except the first, the $\tilde{b}^k$ of step (a) can be set equal to the $b^k \in Bx^k$ computed in step (d) of the previous iteration. Similarly, one can choose $\hat{b}^k = b^k$ is step (c). Combining these two observations, we define a "tight" version of the Douglas-Rachford method:

Definition 3.19. Let $A, B: \mathcal{H} \rightrightarrows \mathcal{H}$ be maximal monotone operators on the Hilbert space $\mathcal{H}$, and $\{\lambda_k\}_{k=0}^{\infty}$ be a sequence of positive scalars. Then the sequence $\{x^k\}_{k=0}^{\infty} \subseteq \operatorname{dom} B$ is said to conform to the *loose Douglas-Rachford recursion* for A, B, and $\{\lambda_k\}$ if

$$\mathbf{x}^{k+1} \in (I + \lambda_k B)^{-1}[(I + \lambda_k A)^{-1}(I - \lambda_k B) + \lambda_k B]\mathbf{x}^k \quad \forall\, k \geq 0 \quad .$$

The sequence $\{(x^k, b^k)\}_{k=0}^{\infty} \subseteq B$ conforms to the *tight Douglas-Rachford recursion* if for all $k \geq 0$, $(x^{k+1}, b^{k+1}) \in B$ is the pair obtained from (x^k, b^k) by the following procedure:

(a') Find the unique $(y^{k+1}, a^{k+1}) \in A$ such that $y^{k+1} + \lambda_k a^{k+1} = x^k - \lambda_k b^k$

(b') Find the unique $(x^{k+1}, b^{k+1}) \in B$ such that $x^{k+1} + \lambda_k b^{k+1} = y^{k+1} + \lambda_k b^k$.

Again, the existence and uniqueness of (y^{k+1}, a^{k+1}) and (x^{k+1}, b^{k+1}) follows from the Representation Lemma, Corollary 3.6.3.

Proposition 3.18. (i) For each $(x^0, b^0) \in B$ and positive stepsize sequence $\{\lambda_k\}_{k=0}^{\infty}$, there exists exactly one sequence $\{(x^k, b^k)\}_{k=0}^{\infty} \subseteq B$ conforming to the tight Douglas-Rachford recursion. (ii) The sequence $\{x^k\}_{k=0}^{\infty} \subseteq \text{dom } B$ defined in this way always conforms to the loose Douglas-Rachford recursion for A, B, and $\{\lambda_k\}$.

Proof. (i) is a direct consequence of the Corollary 3.6.3. Steps (a')-(b') are one possible implementation of (a)-(d) above, which in turn are equivalent to choosing $\mathbf{x}^{k+1} \in (I + \lambda_k B)^{-1}[(I + \lambda_k A)^{-1}(I - \lambda_k B) + \lambda_k B]\mathbf{x}^k$ for all k. Thus, (ii) holds. ■

Figure 12 depicts the Douglas-Rachford iteration; the reader is to assume that the "tight" version of the algorithm is intended unless otherwise noted. Note that, strictly speaking, (x^0, b^0) need not be a member of B; step (b') guarantees $(x^1, b^1) \in B$ for *any* choice of (x^0, b^0).

As with Peaceman-Rachford, all known convergence results for Douglas-Rachford splitting involve the tight version of the algorithm; the loose version is not known to converge for multivalued operators, and is stated mainly for motivational purposes.

As with Peaceman-Rachford, it is possible, when the stepsize parameter is held constant, to express the algorithm solely in terms of the sequence $\{z^k\} = \{x^k + \lambda b^k\}$.

Proposition 3.19. Let $\lambda > 0$ be fixed and let $\{z^k\}_{k=0}^{\infty}$ be a sequence conforming to the recursion

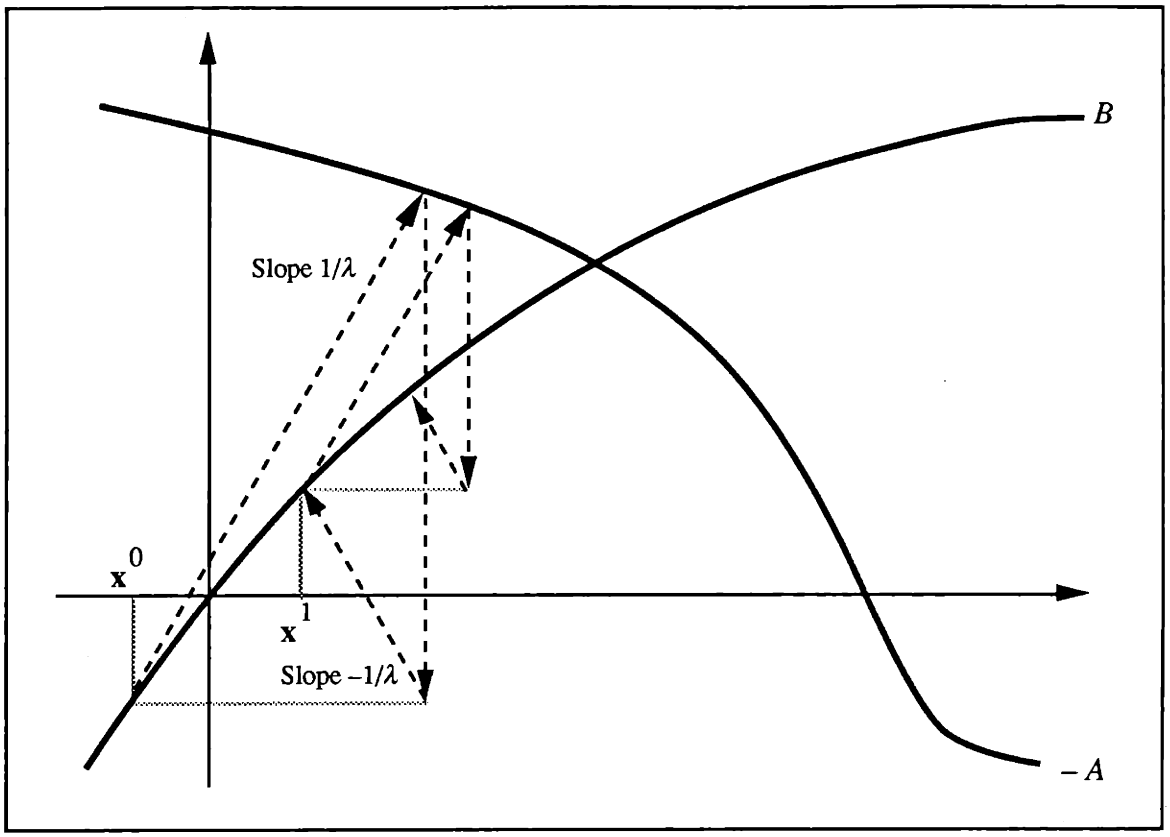


Figure 12. The Douglas-Rachford iteration.

$$z^{k+1} = [J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})]z^k \ .$$

Then the sequence $\{(x^k, b^k)\}_{k=0}^{\infty} \subseteq B$ obtained by letting

$$x^k = J_{\lambda B}\, z^k \qquad b^k = \frac{1}{\lambda}(z^k - x^k)$$

for all $k \geq 0$ is the tight Douglas-Rachford sequence starting at $(x^0, b^0) \in B$, where the stepsize λ_k is held constant at λ.

Proof. We will prove inductively that $z^k = x^k + \lambda b^k$ for all $k \geq 0$, where $\{(x^k, b^k)\}$ is the tight Douglas-Rachford sequence for the constant stepsize λ. Suppose this is true for some given k. Then we have

$$(I - J_{\lambda B})z^k = \lambda b^k \qquad (2J_{\lambda B} - I)z^k = x^k - \lambda b^k \ .$$

Thus,

$$z^{k+1} = [J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})]z^k = J_{\lambda A}(x^k - \lambda b^k) + \lambda b^k = y^k + \lambda b^k \ ,$$

where y^k is as defined in step (a') above. By the description of step (b'), we have

$$z^{k+1} = y^k + \lambda_k b^k = x^{k+1} + \lambda_k b^{k+1} \ .$$

By induction, the claim follows. ■

For any maximal monotone operators A and B, define $G_{\lambda,A,B}$ via

$$G_{\lambda,A,B} = J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B}) \ .$$

The following proposition suggest that the Douglas-Rachford iteration is sensible, in that its fixed points are closely related to the zeroes of $A+B$.

Proposition 3.20. The set of fixed points of the Douglas-Rachford operator $G_{\lambda,A,B} = J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})$ is

$$T_\lambda^* = \{ x + \lambda b \mid (x, b) \in B, (x, -b) \in A \} \ ,$$

the same as for the Peaceman-Rachford operator $(2J_{\lambda A} - I)(2J_{\lambda B} - I)$.

Proof. Consider any point $z \in \mathcal{H}$. Again, z can be written exactly one way as $x + \lambda b$, where $(x, b) \in B$. Then

$$\begin{aligned} & z = [J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})]z \\ \Leftrightarrow \quad & x + \lambda b = J_{\lambda A}(x - \lambda b) + \lambda b \\ \Leftrightarrow \quad & x = J_{\lambda A}(x - \lambda b) \\ \Leftrightarrow \quad & -b \in Ax \\ \Leftrightarrow \quad & z \in T_\lambda^* \ . \end{aligned}$$ ■

The following lemma, from Lions and Mercier (1979), will be stated without proof. A proof and complete discussion, not depending on any further material in this chapter, will be given in Chapter 4 (Corollary 4.2.1).

Lemma 3.2. The Douglas-Rachford operator $G_{\lambda,A,B} = J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})$ is firmly nonexpansive for any maximal monotone operators A and B.

We can now prove the general convergence of the Douglas-Rachford scheme in finite dimension.

Theorem 3.15. Let $A, B: \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ be maximal monotone such that $\operatorname{zer}(A+B) \neq \varnothing$, and let $\{(\mathbf{x}^k, \mathbf{b}^k)\}_{k=0}^{\infty} \subseteq B$ conform to the tight Douglas-Rachford recursion for A, B, and some constant stepsize $\lambda > 0$. Then $\{\mathbf{z}^k\} = \{\mathbf{x}^k + \lambda \mathbf{b}^k\}$ converges to an element of T_λ^*, and $\{\mathbf{x}^k\}$ converges to a zero of $A+B$.

Proof. By Proposition 3.19, $\{\mathbf{z}^k\}$ obeys the recursion

$$\mathbf{z}^{k+1} = [J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})]\mathbf{z}^k \ .$$

Since $J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})$ is firmly nonexpansive, we have (from Theorem 3.10, for instance) that $\{\mathbf{z}^k\}$ converges to a fixed point of $J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})$, if one exists. From Proposition 3.20, such a fixed point exists, and must be an element $\mathbf{z}^*$ of T_λ^*. By the continuity of $J_{\lambda B}$, $\{\mathbf{x}^k\} = \{J_{\lambda B}\mathbf{z}^k\}$ must converge to $J_{\lambda B}\mathbf{z}^*$, which is necessarily a zero of $A+B$. ■

3.5 *Applying Splitting Schemes to Optimization*

In this section, we will study the application of splitting algorithms to optimization. The content will overlap with both Gabay (1983) and Glowinski and Le Tallec (1987), but we will present additional material, and discuss primal splittings as well as dual ones.

Throughout this section, we will consider a generic optimization problem of the form

$$\text{minimize}_{\mathbf{x}\in\mathbb{R}^n} \; f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \quad , \tag{P}$$

where $f\colon \mathbb{R}^n \to (-\infty, +\infty]$ and $g\colon \mathbb{R}^m \to (-\infty, +\infty]$ are both closed proper convex, while $\mathbf{M}$ is some $m \times n$ real matrix. Whenever encountering a problems in the form (P) in the remainder of this thesis, the reader should assume that f and g are both closed proper convex. Solving (P) is equivalent to finding some $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{0} \in \partial[f + g \circ \mathbf{M}]\mathbf{x}$. A very similar problem, in which $-g$ is used in place of g, and g is concave rather than convex, is analyzed in Rockafellar (1970a), Section 31. We essentially repeat much of that analysis here, both because of this minor difference and because of the central importance of the material.

3.5.1 *Primal and Dual Splitting Schemes*

A straightforward way to apply splitting algorithms to (P) is to write

$$\partial[f + g \circ \mathbf{M}] \;=\; \partial f \;+\; \partial[g \circ \mathbf{M}] \quad ,$$

set $A = \partial f$ and $B = \partial[g \circ \mathbf{M}]$, and then apply a splitting method to A and B. Unfortunately, it is not always true that $\partial[f + g \circ \mathbf{M}] \;=\; \partial f \;+\; \partial[g \circ \mathbf{M}]$, as shown by Rockafellar (1970a, 1970c). We make the following definition:

Definition 3.20. The convex program minimize$_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) + g(\mathbf{Mx})$ is *primal splittable* if $\partial[f + g \circ \mathbf{M}] = \partial f + \partial[g \circ \mathbf{M}]$. It is *feasible* if $\inf_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) + g(\mathbf{Mx}) < \infty$, and *bounded* if $\inf_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) + g(\mathbf{Mx}) > -\infty$. It is called *solvable* if it is feasible and has an optimal solution.

Proposition 3.21. Any zero of the operator $\partial f + \partial[g \circ \mathbf{M}]$ is an optimal solution to (P). If (P) is primal splittable, then every optimal solution of (P) is a zero of $\partial f + \partial[g \circ \mathbf{M}]$.

Proof. By Rockafellar (1970a), Theorem 23.8, $\partial f + \partial[g \circ \mathbf{M}] \subseteq \partial[f + g \circ \mathbf{M}]$, so any zero of $\partial f + \partial[g \circ \mathbf{M}]$ is a zero of $\partial[f + g \circ \mathbf{M}]$, and solves (P). If (P) is primal splittable, then we have $\partial[f + g \circ \mathbf{M}] = \partial f + \partial[g \circ \mathbf{M}]$, and any zero of $\partial f + \partial[g \circ \mathbf{M}]$, that is, any solution of (P), is also a zero of $\partial[f + g \circ \mathbf{M}]$. ■

If (P) is not primal splittable, it is possible that (P) has an optimal solution, yet $\partial f + \partial[g \circ \mathbf{M}]$ has no zeroes. In such cases, applying a splitting algorithm to $A = \partial f$ and $B = \partial[g \circ \mathbf{M}]$ would prove futile. Thus, primal splittability is a desirable property, and we will give sufficient conditions that guarantee it. We first need one additional definition and result:

Definition 3.21. A convex function $h: \mathbb{R}^n \to (-\infty, +\infty]$ is called *pseudopolyhedral* if

$$h = \bar{h} + \delta_C \quad ,$$

where $\bar{h}$ is convex with dom $\bar{h} = \mathbb{R}^n$, and C is a nonempty polyhedral convex set.

Proposition 3.22. Any proper polyhedral convex function is pseudopolyhedral.

Proof. Let $h: \mathbb{R}^n \to (-\infty, +\infty]$ be a proper polyhedral convex function. A function is

polyhedral if and only if its epigraph is a polytope. Therefore, epi h is the set of solutions $(\mathbf{x}, y) \in \mathbb{R}^n \times \mathbb{R}$ to some consistent system of linear inequalities

$$\langle \mathbf{a}_i, \mathbf{x} \rangle + c_i y \geq b_i \qquad i = 1,...,s \ ,$$

where none of the c_i are negative, and at least one is positive (otherwise the inequalities would define a polytope that could not possibly be the epigraph of anything). Without loss of generality, we may assume that $c_1 = ... = c_r = 1$ and $c_{r+1} = ... = c_s = 0$ for some r between 1 and s, inclusive. The idea is to separate the "vertical" facets of epi h (those with $c_i = 0$) from the others. Let $\bar{h}$ be the convex function whose epigraph is given by

$$\langle \mathbf{a}_i, \mathbf{x} \rangle + c_i y \geq b_i \qquad i = 1,...,r \ ,$$

namely

$$\bar{h}(\mathbf{x}) = \max_{i=1,...,r} \{b_i - \langle \mathbf{a}_i, \mathbf{x} \rangle\} \ .$$

Let $C \subseteq \mathbb{R}^n$ be the polyhedral set given by the inequality system

$$\langle \mathbf{a}_i, \mathbf{x} \rangle \geq b_i \qquad i = r+1, ...,s.$$

If $r = s$, then we let $C = \mathbb{R}^n$. Then $h = \bar{h} + \delta_C$, and we conclude, since dom $\bar{h} = \mathbb{R}^n$, that h is pseudopolyhedral. ■

We now return to the issue of primal splittability. Even though the matrix $\mathbf{M}$ may not be invertible, or even square, we use the notation $\mathbf{M}^{-1}(S)$ to denote, for any set $S \subseteq \mathbb{R}^n$, the set $\{\mathbf{t} \in \mathbb{R}^n \mid \mathbf{Mt} \in S\}$. Therefore, $\mathrm{dom}(g \circ \mathbf{M}) = \mathbf{M}^{-1}(\mathrm{dom}\ g)$.

Proposition 3.23. The following conditions are each sufficient for a convex program of the form (P) to be primal splittable:

(i) $\text{ri}(\text{dom}\, f) \cap \text{ri}(\mathbf{M}^{-1}(\text{dom}\, g)) \neq \varnothing$.

(ii) f is pseudopolyhedral and $\text{dom}\, f \cap \text{ri}(\mathbf{M}^{-1}(\text{dom}\, g)) \neq \varnothing$.

(iii) g is pseudopolyhedral and $\text{ri}(\text{dom}\, f) \cap \mathbf{M}^{-1}(\text{dom}\, g) \neq \varnothing$.

(iv) Both f and g are pseudopolyhedral and (P) is feasible.

Proof. The sufficiency of (i) is a direct consequence of Rockafellar (1970a), Theorem 23.8. For (ii), we write $f = \bar{f} + \delta_C$, where $\bar{f}$ is convex with domain $\mathbb{R}^n$, and C is polyhedral. Then $\text{dom}\, f = C$, and we have

$$C \cap \text{ri}(\mathbf{M}^{-1}(\text{dom}\, g)) \neq \varnothing \quad \Rightarrow \quad \text{dom}\, \bar{f} \cap C \cap \text{ri}(\mathbf{M}^{-1}(\text{dom}\, g)) \neq \varnothing \quad ,$$

and by the same theorem of Rockafellar,

$$\partial[f + g \circ \mathbf{M}] = \partial[\bar{f} + \delta_C + g \circ \mathbf{M}] = \partial(\bar{f}) + \partial\delta_C + \partial[g \circ \mathbf{M}] = \partial f + \partial[g \circ \mathbf{M}] \quad .$$

For (iii), we write $g = \bar{g} + \delta_D$, where $\bar{g}$ is convex and finite on $\mathbb{R}^m$, and $D \subseteq \mathbb{R}^m$ is polyhedral. Then $g \circ \mathbf{M} = \bar{g} \circ \mathbf{M} + \delta_{\mathbf{M}^{-1}(D)}$. Now, $\bar{g} \circ \mathbf{M}$ is finite and convex on $\mathbb{R}^n$, and $\mathbf{M}^{-1}(D)$ is polyhedral, so $g \circ \mathbf{M}$ is pseudopolyhedral. The proof then proceeds much as for (ii). The proof for (iv) is a combination of the proofs of (ii) and (iii). ■

Any splitting algorithm must execute some combination of forward and proximal steps on the operators A and B which constitute the splitting. We indicate how to perform such calculations in the case that $A = \partial f$ and $B = \partial[g \circ \mathbf{M}]$.

Proposition 3.24. Consider a convex program in the form (P).

(i) Given some $\bar{\mathbf{x}} \in \text{dom}\, f$, the forward step $\mathbf{x}' \in (I - \lambda\partial f)\bar{\mathbf{x}}$ on the operator ∂f may be performed by the calculation

Choose any $\mathbf{y} \in \partial f(\overline{\mathbf{x}})$
$\mathbf{x}' = \overline{\mathbf{x}} - \lambda \mathbf{y}$.

(ii) Given some $\overline{\mathbf{x}} \in \text{dom}(g \circ \mathbf{M})$, the forward step $\mathbf{x}' \in (I - \lambda \partial[g \circ \mathbf{M}])\overline{\mathbf{x}}$ on the operator $\partial[g \circ \mathbf{M}]$ may be performed by the calculation

Choose any $\mathbf{y} \in \partial g(\mathbf{M}\overline{\mathbf{x}})$
$\mathbf{x}' = \overline{\mathbf{x}} - \lambda \mathbf{M}^{\mathsf{T}}\mathbf{y}$.

(iii) Given some $\overline{\mathbf{x}} \in \mathbb{R}^n$, the proximal step $\mathbf{x}' \in (I + \lambda \partial f)^{-1}\overline{\mathbf{x}}$ on the operator ∂f may be performed by the calculation

$$\mathbf{x}' = \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{x} - \overline{\mathbf{x}}\|^2\} \quad .$$

(iv) Given some $\overline{\mathbf{x}} \in \mathbb{R}^n$, the proximal step $\mathbf{x}' \in (I + \lambda \partial[g \circ \mathbf{M}])^{-1}\overline{\mathbf{x}}$ on the operator $\partial[g \circ \mathbf{M}]$ may be performed by the calculation

$$\mathbf{x}' = \arg\min_{\mathbf{x}} \{g(\mathbf{M}\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{x} - \overline{\mathbf{x}}\|^2\} \quad .$$

Proof. (i) requires no explanation. By Rockafellar (1970a), Theorem 23.9,

$$\partial[g \circ \mathbf{M}] \supseteq \mathbf{M}^{\mathsf{T}} \circ \partial g \circ \mathbf{M} \ ,$$

which suffices to establish (ii). For (iii), one has

$$\begin{aligned}
& \mathbf{x}' \in (I + \lambda \partial f)^{-1}\overline{\mathbf{x}} \\
\Leftrightarrow \quad & (I + \lambda \partial f)\mathbf{x}' \ni \overline{\mathbf{x}} \\
\Leftrightarrow \quad & \lambda \partial f(\mathbf{x}') \ni \overline{\mathbf{x}} - \mathbf{x}' \\
\Leftrightarrow \quad & \partial f(\mathbf{x}') \ni \frac{1}{\lambda}(\overline{\mathbf{x}} - \mathbf{x}') \\
\Leftrightarrow \quad & \mathbf{0} \in \partial f(\mathbf{x}') + \frac{1}{\lambda}(\mathbf{x}' - \overline{\mathbf{x}}) \ .
\end{aligned}$$

Let $\widehat{f}(\mathbf{x}) = f(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{x} - \overline{\mathbf{x}}\|^2$. By conventional calculus and Theorem 23.8 of Rockafellar (1970a) we have that

$$\partial \widehat{f}(\mathbf{x}) = \partial f(\mathbf{x}) + \frac{1}{\lambda}(\mathbf{x} - \overline{\mathbf{x}}) \ .$$

Note that since f is proper, $\widehat{f}(\mathbf{x})$ is proper, and furthermore, since $\widehat{f}(\mathbf{x})$ is strongly convex, $\widehat{f}(\mathbf{x})$ has a unique global minimum. Then

$$\mathbf{0} \in \partial f(\mathbf{x}') + \tfrac{1}{\lambda}(\mathbf{x}' - \overline{\mathbf{x}}) \;\Leftrightarrow\; \mathbf{0} \in \partial \widehat{f}(\mathbf{x}') \;\Leftrightarrow\; \mathbf{x}' = \arg\min_{\mathbf{x}} \{ \widehat{f}(\mathbf{x}) \} \ ,$$

establishing (iii). The proof of (iv) is similar. ■

We now introduce a duality framework for the problem (P) by writing it in the form

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} & \mathbf{M}\mathbf{x} = \mathbf{z} \ , \end{array} \qquad \text{(P')}$$

and attaching a multiplier vector the constraints $\mathbf{M}\mathbf{x} = \mathbf{z}$.

For any function $h{:}\mathbb{R}^n \to (-\infty, +\infty]$, let h^* denote its convex conjugate (*e.g.* Rockafellar 1970a)

$$h^*(\mathbf{y}) \;=\; \sup_{\mathbf{x}} \{ \langle \mathbf{x}, \mathbf{y} \rangle \;-\; h(\mathbf{x}) \} \ .$$

One can form a problem dual to (P') by writing

$$\text{maximize}_{\ \mathbf{p} \in \mathbb{R}^n} \ q(\mathbf{p}) \ ,$$

where

$$\begin{aligned} q(\mathbf{p}) \;&=\; \inf_{\mathbf{x},\,\mathbf{z}} \{ f(\mathbf{x}) + g(\mathbf{z}) + \langle \mathbf{p}, \mathbf{M}\mathbf{x} - \mathbf{z} \rangle \} \} \\ &=\; \inf_{\mathbf{x}} \{ f(\mathbf{x}) + \langle \mathbf{p}, \mathbf{M}\mathbf{x} \rangle \} \;+\; \inf_{\mathbf{z}} \{ g(\mathbf{z}) - \langle \mathbf{p}, \mathbf{z} \rangle \} \\ &=\; -f^*(-\mathbf{M}^{\mathrm{T}}\mathbf{p}) \;-\; g^*(\mathbf{p}) \ . \end{aligned}$$

Thus the dual problem, maximize $_{\mathbf{p} \in \mathbb{R}^n}\ q(\mathbf{p})$, may be written

$$\text{maximize}_{\ \mathbf{p} \in \mathbb{R}^n} \ -f^*(-\mathbf{M}^{\mathrm{T}}\mathbf{p}) \;-\; g^*(\mathbf{p}) \quad ,$$

or, equivalently,

$$\text{minimize}_{\,\mathbf{p} \in \mathbb{R}^n} \; f^*(-\mathbf{M}^T\mathbf{p}) + g^*(\mathbf{p}) \quad . \tag{D}$$

(D) is the form of the dual problem we will customarily use. Note its resemblance to (P).

We will now develop some fundamental properties of the duality relationship that exists between (P) and (D). Before exploring this structure, we need some important facts from the theory of convex conjugate functions, all of which are proved in Rockafellar (1970a), Chapter 23.

Theorem 3.16. Let $h: \mathbb{R}^n \to (-\infty, +\infty]$ be closed, proper, and convex. Then for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,

$$h(\mathbf{x}) + h^*(\mathbf{y}) \geq \langle \mathbf{x}, \mathbf{y} \rangle \ ,$$

and furthermore, the following three statements are equivalent:

(i) $h(\mathbf{x}) + h^*(\mathbf{y}) = \langle \mathbf{x}, \mathbf{y} \rangle$

(ii) $(\mathbf{x}, \mathbf{y}) \in \partial h$

(iii) $(\mathbf{y}, \mathbf{x}) \in \partial h^*$.

Thus, $\partial h^* = (\partial h)^{-1}$.

We can now obtain a *weak* duality relation between (P) and (D):

Proposition 3.25 (Weak Duality). For any $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{p} \in \mathbb{R}^m$,

$$f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \geq -\left(f^*(-\mathbf{M}^T\mathbf{p}) + g^*(\mathbf{p})\right) .$$

Proof. From Theorem 3.16, we have that

$$f(\mathbf{x}) + f^*(-\mathbf{M}^T\mathbf{p}) \geq \langle \mathbf{x}, -\mathbf{M}^T\mathbf{p} \rangle = -\langle \mathbf{Mx}, \mathbf{p} \rangle$$

$$g(\mathbf{Mx}) + g^*(\mathbf{p}) \geq \langle \mathbf{Mx}, \mathbf{p} \rangle ,$$

and therefore

$$f(\mathbf{x}) + f^*(-\mathbf{M}^T\mathbf{p}) \geq -\langle \mathbf{Mx}, \mathbf{p} \rangle \geq -\big(g(\mathbf{Mx}) + g^*(\mathbf{p})\big) .$$

Rearranging $f(\mathbf{x}) + f^*(-\mathbf{M}^T\mathbf{p}) \geq -\big(g(\mathbf{Mx}) + g^*(\mathbf{p})\big)$, one obtains

$$f(\mathbf{x}) + g(\mathbf{Mx}) \geq -\big(f^*(-\mathbf{M}^T\mathbf{p}) + g^*(\mathbf{p})\big) . \quad \blacksquare$$

Corollary 3.25.1. If (P) is unbounded, then (D) must be infeasible. If (D) is unbounded, then (P) must be infeasible.

Proposition 3.26 (A Kuhn-Tucker Theorem). The following four conditions are equivalent to one another, and each imply that $\mathbf{x} \in \mathbb{R}^n$ is optimal for (P) while $\mathbf{p} \in \mathbb{R}^m$ is simultaneously optimal for (D):

(i) $\mathbf{x}$ is optimal for (P) and $\mathbf{p}$ is such that $\mathbf{p} \in \partial g(\mathbf{Mx})$ and $-\mathbf{M}^T\mathbf{p} \in \partial f(\mathbf{x})$.

(ii) $\mathbf{p}$ is optimal for (D) and $\mathbf{x}$ is such that $\mathbf{x} \in \partial f^*(-\mathbf{M}^T\mathbf{p})$ and $\mathbf{Mx} \in \partial g^*(\mathbf{p})$.

(iii) $(\mathbf{x}, -\mathbf{M}^T\mathbf{p}) \in \partial f$ and $(\mathbf{Mx}, \mathbf{p}) \in \partial g$.

(iv) $f(\mathbf{x}) + g(\mathbf{Mx}) = -\big(f^*(-\mathbf{M}^T\mathbf{p}) + g^*(\mathbf{p})\big)$.

Proof. We start by showing that (i)-(iv) are equivalent to one another. First, (iii) is just a restatement of (i) without the stipulation that $\mathbf{x}$ be optimal, so (i) $\Rightarrow$ (iii). To show (iii) $\Rightarrow$ (i), we need to show that $\mathbf{x}$ must be optimal for (P). Since $\mathbf{M}^T \circ \partial g \circ \mathbf{M} \subseteq \partial[g \circ \mathbf{M}]$, one has

$$\mathbf{M}^T\mathbf{p} \in \partial[g \circ \mathbf{M}](\mathbf{x}) \quad \Rightarrow \quad 0 = -\mathbf{M}^T\mathbf{p} + \mathbf{M}^T\mathbf{p} \in \partial f(\mathbf{x}) + \partial[g \circ \mathbf{M}](\mathbf{x}) \subseteq \partial[f + g \circ \mathbf{M}]\mathbf{x} ,$$

and so $\mathbf{x}$ is optimal for (P). Since $\partial h^* = (\partial h)^{-1}$ for any closed proper convex function h, we note that (iii) is equivalent to

$$\text{(iii')} \qquad (-\mathbf{M}^T\mathbf{p}, \mathbf{x}) \in \partial f^* \text{ and } (\mathbf{p}, \mathbf{M}\mathbf{x}) \in \partial g^*.$$

The proof that (iii') $\Leftrightarrow$ (ii) may now proceed in an analogous manner to (iii) $\Leftrightarrow$ (i).

We now show that (iii) $\Leftrightarrow$ (iv). First, (iii) is equivalent to

$$f(\mathbf{x}) + f^*(-\mathbf{M}^T\mathbf{p}) = \langle \mathbf{x}, -\mathbf{M}^T\mathbf{p} \rangle = -\langle \mathbf{M}\mathbf{x}, \mathbf{p} \rangle$$

$$g(\mathbf{M}\mathbf{x}) + g^*(\mathbf{p}) = \langle \mathbf{M}\mathbf{x}, \mathbf{p} \rangle \quad .$$

Adding these two equations and rearranging, one obtains (iv). Conversely, (iv) implies

$$f(\mathbf{x}) + f^*(-\mathbf{M}^T\mathbf{p}) = -\big(g(\mathbf{M}\mathbf{x}) + g^*(\mathbf{p})\big) \ ,$$

which, in view of $f(\mathbf{x}) + f^*(-\mathbf{M}^T\mathbf{p}) \geq -\langle \mathbf{M}\mathbf{x}, \mathbf{p} \rangle \geq -\big(g(\mathbf{M}\mathbf{x}) + g^*(\mathbf{p})\big)$, means that

$$f(\mathbf{x}) + f^*(-\mathbf{M}^T\mathbf{p}) = -\langle \mathbf{M}\mathbf{x}, \mathbf{p} \rangle$$

$$g(\mathbf{M}\mathbf{x}) + g^*(\mathbf{p}) = \langle \mathbf{M}\mathbf{x}, \mathbf{p} \rangle \quad .$$

By Theorem 3.16, we obtain (iii).

Finally, in view of our weak duality result, (iv) implies that $\mathbf{x}$ and $\mathbf{p}$ are optimal for (P) and (D), respectively. ■

A pair $(\mathbf{x}, \mathbf{p})$ such that $(\mathbf{x}, -\mathbf{M}^T\mathbf{p}) \in \partial f$ and $(\mathbf{M}\mathbf{x}, \mathbf{p}) \in \partial g$ is called a *Kuhn-Tucker pair* for (P).

Proposition 3.27. The existence of a Kuhn-Tucker pair $(\mathbf{x}, \mathbf{p})$ is sufficient to imply that $\text{zer}(\partial f + \partial[g \circ \mathbf{M}])$ and $\text{zer}(\partial[f^* \circ (-\mathbf{M}^T)] + \partial g^*)$ are nonempty.

Proof. By Rockafellar (1970a), Theorem 23.9, Kuhn-Tucker condition (i) implies $\mathbf{M}^T\mathbf{p} \in \partial[g \circ \mathbf{M}](\mathbf{x})$ and $-\mathbf{M}^T\mathbf{p} \in \partial f(\mathbf{x})$, so $\text{zer}(\partial f + \partial[g \circ \mathbf{M}])$ is nonempty. The dual statement follows similarly from condition (ii). ■

We have already mentioned that it is possible to apply splitting algorithms to optimization problems in the form (P) by letting $A = \partial f$ and $B = \partial[g \circ \mathbf{M}]$, and then applying a splitting algorithm to A and B. Another approach is to attempt to solve the dual problem (D), which is equivalent to finding a zero of $\partial[f^* \circ (-\mathbf{M}^T) + g^*]$, by some form of splitting. It is natural in this case is to let $A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$, and apply a splitting algorithm to A and B. Because of the intimate relationship between (P) and (D) demonstrated above, such approaches will generally also yield an optimal vector for (P), as well. To be able to use such *dual splitting* algorithms, one needs a dual property akin to primal splittability, which we will now introduce:

Definition 3.22. The convex program (P) is called *dual splittable if*

$$\partial[f^* \circ (-\mathbf{M}^T) + g^*] = \partial[f^* \circ (-\mathbf{M}^T)] + \partial g^* .$$

It is called *primal normal* if $\partial[g \circ \mathbf{M}] = \mathbf{M}^T \circ \partial g \circ \mathbf{M}$ and said to be *dual normal if* $\partial[f^* \circ (-\mathbf{M}^T)] = -\mathbf{M} \circ \partial f^* \circ (-\mathbf{M}^T)$.

Dual splittability, or at least the existence of a Kuhn-Tucker pair, is very important for the viability of dual splitting algorithms, for otherwise it is possible that $\partial[f^* \circ (-\mathbf{M}^T)] + \partial g^*$ could have no zeroes, even though $\partial[f^* \circ (-\mathbf{M}^T) + g^*]$ does. In such cases, applying a splitting algorithm to $A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$ would be of little use.

We now give sufficient conditions for dual splittability.

Proposition 3.28. The following conditions are each sufficient for (P) to be dual splittable.

(i) $\text{ri}(\text{im}\,\partial g) \cap \text{ri}((-\mathbf{M}^{\mathsf{T}})^{-1}(\text{im}\,\partial f)) \neq \varnothing$.

(ii) g^* is pseudopolyhedral and $(\text{im}\,\partial g) \cap \text{ri}((-\mathbf{M}^{\mathsf{T}})^{-1}(\text{im}\,\partial f)) \neq \varnothing$.

(iii) f^* is pseudopolyhedral and $\text{ri}(\text{im}\,\partial g) \cap (-\mathbf{M}^{\mathsf{T}})^{-1}(\text{im}\,\partial f) \neq \varnothing$.

(iv) f^* and g^* are both pseudopolyhedral and (D) is feasible.

Proof. First, note that the problem (D), minimize $_{\mathbf{p}\in\mathbb{R}^m}\ f^*(-\mathbf{M}^{\mathsf{T}}\mathbf{p}) + g^*(\mathbf{p})$, is in essentially the same form as (P). Applying Proposition 3.23(i)-(iv) and using the identities

$$\text{dom}\,\partial g^* \;=\; \text{dom}((\partial g)^{-1}) \;=\; \text{im}\,\partial g$$

$$\text{dom}\,\partial f^* \;=\; \text{dom}((\partial f)^{-1}) \;=\; \text{im}\,\partial f$$

proves that (i)-(iv) are each sufficient. ■

Also note that for f^* or g^* to be pseudopolyhedral, it is sufficient to have f or g polyhedral, respectively. This is because the conjugate of a polyhedral function is polyhedral, and therefore pseudopolyhedral.

We now give some conditions guaranteeing primal and dual normality. First, we need to demonstrate some distributive laws for the algebra of multivalued mappings.

Lemma 3.3. Let $A,B\colon \mathcal{X} \rightrightarrows \mathcal{X}$ be any mapping on a vector space $\mathcal{X}$. Let Q be a single-valued function $\mathcal{X} \to \mathcal{X}$. Then the following right distributive law holds:

$$(A + B)Q = AQ + BQ \ .$$

If Q is linear, as well as single-valued, then one also has the left distributive identity

$$Q(A + B) = QA + QB \ .$$

Proof. For the right distributive law, we have

$$\begin{aligned} AQ + BQ &= \{(w, a+b) \mid (w, a) \in AQ, (w, b) \in BQ\} \\ &= \{(w, a+b) \mid \exists\, x,x' \colon x \in Q(w), x' \in Q(w),\ (x, a) \in A, (x', b) \in B\} \\ &= \{(w, a+b) \mid \exists\, x \colon x \in Q(w), (x, a) \in A, (x, b) \in B\} \\ &= (A + B)Q \ . \end{aligned}$$

For the left law, using the assumption that Q is linear,

$$\begin{aligned} Q(A + B) &= \{(x, Q(a+b)) \mid a \in Ax, b \in Bx\} \\ &= \{(x,\ Qa + Qb) \mid a \in Ax, b \in Bx\} \\ &= \{(x, Qa) \mid a \in Ax\} + \{(x, Qb) \mid b \in Bx\} \\ &= QA + QB \ . \ \blacksquare \end{aligned}$$

Proposition 3.29. The following conditions are each sufficient to ensure that (P) is primal normal:

(i) $\operatorname{im} \mathbf{M} \cap \operatorname{ri}(\operatorname{dom} g) \neq \varnothing$.

(ii) g is pseudopolyhedral and $\operatorname{im} \mathbf{M} \cap \operatorname{dom} g \neq \varnothing$.

Proof. The sufficiency of (i) follows directly from Rockafellar (1970a), Theorem 23.9. For (ii), we write $g = \overline{g} + \delta_D$, where $\overline{g}$ is convex and finite-valued, while $D \neq \varnothing$ is polyhedral.

Then, from the distributive laws for the algebra of mappings and Theorems 23.8 and 23.9 of Rockafellar,

$$\begin{aligned}\partial[g \circ \mathbf{M}] &= \partial[\,\bar{g} \circ \mathbf{M} + \delta_D \circ \mathbf{M}\,] \\ &= (\mathbf{M}^T \circ \partial\bar{g} \circ \mathbf{M}) + (\mathbf{M}^T \circ \partial\delta_D \circ \mathbf{M}) \\ &= \mathbf{M}^T(\partial g \circ \mathbf{M} + \partial\delta_D \circ \mathbf{M}) \\ &= \mathbf{M}^T \circ \partial g \circ \mathbf{M} \quad . \qquad \blacksquare\end{aligned}$$

Proposition 3.30. The following conditions are each sufficient to ensure that (P) is dual normal:

(i) $\operatorname{im} \mathbf{M}^T \cap \operatorname{ri}(\operatorname{im} \partial f) \neq \varnothing$.

(ii) f^* is pseudopolyhedral and $\operatorname{im} \mathbf{M}^T \cap \operatorname{im} \partial f \neq \varnothing$.

Proof. We first note that $\operatorname{im} \mathbf{M}^T = \operatorname{im} (-\mathbf{M}^T)$, and that $\operatorname{dom} \partial f^* = \operatorname{im} \partial f$. The proof then proceeds as for Proposition 3.29. ■

We are now also able to state some conditions under which the "duality gap" between (P) and (D) vanishes, and both have solutions.

Proposition 3.31. The following conditions are each sufficient to guarantee that

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) = -\min_{\mathbf{p} \in \mathbb{R}^m} f^*(-\mathbf{M}^T\mathbf{p}) + g^*(\mathbf{p})$$

and that there exists a Kuhn-Tucker pair $(\mathbf{x}, \mathbf{p})$:

(i) (P) is feasible, primal splittable, and primal normal, and solvable.

(ii) (P) is dual splittable and dual normal, and (D) is solvable.

Proof. We first examine (i). Let $\mathbf{x}$ be an optimal solution to (P), hence $\partial[f + g \circ \mathbf{M}]\mathbf{x} \ni \mathbf{0}$. By primal splittability and normality, $\partial[f + g \circ \mathbf{M}] = \partial f + (\mathbf{M}^T \circ \partial g \circ \mathbf{M})$, hence

$$\mathbf{0} \in \partial f(\mathbf{x}) + \mathbf{M}^T \partial g(\mathbf{M}\mathbf{x}) \ .$$

Therefore, there exists some $\mathbf{p} \in \partial g(\mathbf{M}\mathbf{x})$ such that $-\mathbf{M}^T\mathbf{p} \in \partial f(\mathbf{x})$. This is precisely Kuhn-Tucker condition of Proposition 3.26(i), so that the desired conclusion follows from Proposition 3.26. The proof for condition (ii) is similar. ■

To apply splitting algorithms to the operators $A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$, one must be able to compute the forward and proximal steps for these operators. We now show how such calculations may be accomplished.

Proposition 3.32. Consider any convex program in the form (P).

(i) Consider any $\overline{\mathbf{p}} \in \operatorname{im} \partial g$. Then the forward step $\mathbf{p}' = (I - \lambda \partial g^*)\overline{\mathbf{p}}$ on the operator ∂g^* may be performed via the procedure

Let θ be any vector in $\mathbb{R}^m$
Choose $\overline{\mathbf{z}} \in \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \overline{\mathbf{p}}, \mathbf{z} - \theta \rangle\}$
$\mathbf{p}' = \overline{\mathbf{p}} - \lambda \overline{\mathbf{z}}$.

(ii) Consider any $\overline{\mathbf{p}}$ such that $\overline{\mathbf{p}} \in \operatorname{dom}(\partial f^* \circ (-\mathbf{M}^T))$, that is, $-\mathbf{M}^T\overline{\mathbf{p}} \in \operatorname{im} \partial f$. Then the forward step $\mathbf{p}' = (I - \lambda \partial[f^* \circ (-\mathbf{M}^T)])\overline{\mathbf{p}}$ on the operator $\partial[f^* \circ (-\mathbf{M}^T)]$ may be performed via

Let ϕ be any vector in $\mathbb{R}^m$
Choose $\overline{\mathbf{x}} \in \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \overline{\mathbf{p}}, \mathbf{M}\mathbf{x} - \phi \rangle\}$
$\mathbf{p}' = \overline{\mathbf{p}} + \lambda \mathbf{M}\overline{\mathbf{x}}$.

(iii) Consider any $\bar{\mathbf{p}} \in \mathbb{R}^m$. Then the proximal step $\mathbf{p}' = (I + \lambda \partial g^*)^{-1}\bar{\mathbf{p}}$ on the operator ∂g^* may be performed via the procedure

Let θ, ζ be any vectors in $\mathbb{R}^m$

$$\bar{\mathbf{z}} = \arg\min_{\mathbf{z}} \{ g(\mathbf{z}) - \langle \bar{\mathbf{p}} - \lambda\zeta, \mathbf{z} - \theta \rangle + \frac{\lambda}{2} \| \mathbf{z} - \zeta \|^2 \}$$

$$\mathbf{p}' = \bar{\mathbf{p}} - \lambda \bar{\mathbf{z}} \ .$$

Furthermore, $\bar{\mathbf{z}} \in \partial g^*(\mathbf{p}')$.

(iv) Consider any $\bar{\mathbf{p}} \in \mathbb{R}^m$, and further assume that $\mathbf{M}$ has full column rank, that is, that the null space of $\mathbf{M}$ is $\{\mathbf{0}\}$. Then the proximal step $\mathbf{p}' = (I + \lambda \partial [f^* \circ (-\mathbf{M}^T)])^{-1}\bar{\mathbf{p}}$ on the operator $\partial [f^* \circ (-\mathbf{M}^T)]$ may be performed via

Let η, ϕ be any vectors in $\mathbb{R}^m$

$$\bar{\mathbf{x}} = \arg\min_{\mathbf{x}} \{ f(\mathbf{x}) + \langle \bar{\mathbf{p}} + \lambda\eta, \mathbf{M}\mathbf{x} - \phi \rangle + \frac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \eta \|^2 \}$$

$$\mathbf{p}' = \bar{\mathbf{p}} + \lambda \mathbf{M} \bar{\mathbf{x}} \ .$$

Furthermore, $(\bar{\mathbf{x}}, -\mathbf{M}^T\mathbf{p}') \in \partial f$ and $-\mathbf{M}\bar{\mathbf{x}} \in \partial [f^* \circ (-\mathbf{M}^T)](\mathbf{p}')$.

Proof. For (i), note that since $\bar{\mathbf{p}} \in \text{im}\, \partial g$, there exists some $\bar{\mathbf{z}} \in \mathbb{R}^m$ such that $\partial g(\bar{\mathbf{z}}) \ni \bar{\mathbf{p}}$, hence, using "$\partial_{\mathbf{z}}$" to denote the subgradient with respect to $\mathbf{z}$,

$$\partial_{\mathbf{z}} [g(\mathbf{z}) - \langle \bar{\mathbf{p}}, \mathbf{z} - \theta \rangle] = \partial_{\mathbf{z}} [g(\mathbf{z}) - \langle \bar{\mathbf{p}}, \mathbf{z} \rangle] \ni \mathbf{0} \ .$$

So, $\bar{\mathbf{z}}$ solves the given minimization problem (note that the constant term $\langle \bar{\mathbf{p}}, \theta \rangle$ is irrelevant). We then have $(\bar{\mathbf{z}}, \bar{\mathbf{p}}) \in \partial g$, hence $(\bar{\mathbf{p}}, \bar{\mathbf{z}}) \in \partial g^*$, and $\mathbf{p}' = \bar{\mathbf{p}} - \lambda\bar{\mathbf{z}}$ is a valid forward step.

The proof of (ii) is similar to (i), except for the appearance of the matrix $\mathbf{M}$. Since $-\mathbf{M}^T\bar{\mathbf{p}} \in \text{im}\, \partial f, f(\mathbf{x}) + \langle \bar{\mathbf{p}}, \mathbf{M}\mathbf{x} - \phi \rangle$, considered as a function of $\mathbf{x}$, has at least one minimizer $\bar{\mathbf{x}}$. For such a point, $(\bar{\mathbf{x}}, -\mathbf{M}^T\bar{\mathbf{p}}) \in \partial f$, and so $(-\mathbf{M}^T\bar{\mathbf{p}}, \bar{\mathbf{x}}) \in \partial f^*$. Since

$$-\mathbf{M} \circ \partial f^* \circ (-\mathbf{M}^T) \subseteq \partial[f^* \circ (-\mathbf{M}^T)] \ ,$$

one has

$$-\mathbf{M}\bar{\mathbf{x}} \in -\mathbf{M}\partial f^*(-\mathbf{M}^T\bar{\mathbf{p}}) = [-\mathbf{M} \circ \partial f \circ (-\mathbf{M}^T)]\bar{\mathbf{p}} \subseteq \partial[f^* \circ (-\mathbf{M}^T)]\bar{\mathbf{p}} \ ,$$

and so

$$\mathbf{p}' = \bar{\mathbf{p}} - \lambda(-\mathbf{M}\bar{\mathbf{x}}) = \bar{\mathbf{p}} + \lambda\mathbf{M}\bar{\mathbf{x}}$$

is a correct calculation of $\mathbf{p}'$.

To show (iii), we note, by the convexity and properness of g and the presence of the quadratic term $\frac{\lambda}{2}\|\mathbf{z} - \zeta\|^2$, that $g(\mathbf{z}) - \langle \bar{\mathbf{p}} - \lambda\zeta, \mathbf{z} - \theta \rangle + \frac{\lambda}{2}\|\mathbf{z} - \zeta\|^2$ must be a proper, strongly convex function of $\mathbf{z}$, and therefore must possess a unique global minimizer $\bar{\mathbf{z}}$. We then have

$$\begin{aligned}
& \mathbf{0} \in \partial_{\mathbf{z}}[g(\mathbf{z}) - \langle \bar{\mathbf{p}} - \lambda\zeta, \mathbf{z} - \theta \rangle + \tfrac{\lambda}{2}\|\mathbf{z} - \zeta\|^2] \Big|_{\mathbf{z}=\bar{\mathbf{z}}} \\
\Leftrightarrow \quad & \mathbf{0} \in \partial g(\bar{\mathbf{z}}) - \bar{\mathbf{p}} - \lambda\zeta + \lambda(\bar{\mathbf{z}} - \zeta) \\
\Leftrightarrow \quad & \mathbf{0} \in \partial g(\bar{\mathbf{z}}) - \bar{\mathbf{p}} + \lambda\bar{\mathbf{z}} \\
\Leftrightarrow \quad & (\bar{\mathbf{z}}, \bar{\mathbf{p}} - \lambda\bar{\mathbf{z}}) = (\bar{\mathbf{z}}, \mathbf{p}') \in \partial g \\
\Leftrightarrow \quad & (\mathbf{p}', \bar{\mathbf{z}}) = (\mathbf{p}', \tfrac{1}{\lambda}(\bar{\mathbf{p}} - \mathbf{p}')) \in \partial g^* \\
\Leftrightarrow \quad & \bar{\mathbf{p}} - \mathbf{p}' \in \lambda\partial g^*(\mathbf{p}') \\
\Leftrightarrow \quad & \mathbf{p}' + \lambda\partial g^*(\mathbf{p}') \ni \bar{\mathbf{p}} \\
\Leftrightarrow \quad & \mathbf{p}' = (I + \lambda\partial g^*)^{-1}\bar{\mathbf{p}} \ ,
\end{aligned}$$

where we have twice used that $\mathbf{p}' = \bar{\mathbf{p}} - \lambda\bar{\mathbf{z}}$.

The proof of (iv) resembles that of (iii), except for the appearance of the matrix $\mathbf{M}$. The assumption that $\mathbf{M}$ has full rank implies, along with f being proper convex, that $f(\mathbf{x}) + \langle \bar{\mathbf{p}} + \lambda\eta, \mathbf{M}\mathbf{x} - \phi \rangle + \frac{\lambda}{2}\|\mathbf{M}\mathbf{x} - \eta\|^2$ is a proper and strongly convex function of $\mathbf{x}$, and

hence possesses a unique minimizer $\bar{\mathbf{x}}$. Then, using "$\partial_{\mathbf{x}}$" to denote the subgradient with respect to $\mathbf{x}$,

$$
\begin{aligned}
& \mathbf{0} \in \partial_{\mathbf{x}}[f(\mathbf{x}) + \langle \bar{\mathbf{p}} + \lambda\eta, \mathbf{M}\mathbf{x} - \phi \rangle + \tfrac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \eta \|^2] \Big|_{\mathbf{x}=\bar{\mathbf{x}}} \\
\Leftrightarrow \quad & \mathbf{0} \in \partial f(\bar{\mathbf{x}}) + \mathbf{M}^T(\bar{\mathbf{p}} + \lambda\eta) + \lambda \mathbf{M}^T(\mathbf{M}\bar{\mathbf{x}} - \eta) \\
\Leftrightarrow \quad & \mathbf{0} \in \partial f(\bar{\mathbf{x}}) + \mathbf{M}^T\bar{\mathbf{p}} + \lambda \mathbf{M}^T\mathbf{M}\bar{\mathbf{x}} = \partial f(\bar{\mathbf{x}}) + \mathbf{M}^T\mathbf{p}' \\
\Leftrightarrow \quad & (\bar{\mathbf{x}}, -\mathbf{M}^T\mathbf{p}') \in \partial f \\
\Leftrightarrow \quad & (-\mathbf{M}^T\mathbf{p}', \bar{\mathbf{x}}) \in \partial f^* \\
\Rightarrow \quad & (\mathbf{p}', \bar{\mathbf{x}}) \in \partial f^* \circ (-\mathbf{M}^T) \\
\Rightarrow \quad & (\mathbf{p}', -\mathbf{M}\bar{\mathbf{x}}) \in -\mathbf{M} \circ \partial f \circ (-\mathbf{M}^T) \subseteq \partial[f^* \circ (-\mathbf{M}^T)] \\
\Rightarrow \quad & (\mathbf{p}', \tfrac{1}{\lambda}(\bar{\mathbf{p}} - \mathbf{p}')) \in \partial[f^* \circ (-\mathbf{M}^T)] \\
\Leftrightarrow \quad & \mathbf{p}' = (I + \lambda\partial[f^* \circ (-\mathbf{M}^T)])^{-1}\bar{\mathbf{p}} \ ,
\end{aligned}
$$

where we have used $\mathbf{p}' = \bar{\mathbf{p}} + \lambda\mathbf{M}\bar{\mathbf{x}}$ twice. ■

Note that the assumption that $\mathbf{M}$ has full column rank in (iv) is needed only to guarantee the existence of $\bar{\mathbf{x}}$. If $\mathbf{M}$ is rank-deficient but an $\bar{\mathbf{x}}$ minimizing $f(\mathbf{x}) + \langle \bar{\mathbf{p}} + \lambda\eta, \mathbf{M}\mathbf{x} - \phi \rangle + \frac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \eta \|^2$ happens to exist, the result $\mathbf{p}'$ of the proximal step on $\partial[f^* \circ (-\mathbf{M}^T)]$ may still be computed as indicated in (iv). From now on, however, we will generally assume that $\mathbf{M}$ has full column rank.

The inclusion of the arbitrary, apparently irrelevant, and self-cancelling terms ζ, θ, ϕ, and η in the above proposition may seem somewhat pointless. However, they will prove convenient, in the following sections, for relating the forward and backward steps on $A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$ to various operations on the (augmented) Lagrangian of the problem (P'). In general, we can already remark that dual forward steps involve minimizing functions resembling Lagrangians, whereas dual proximal steps involve minimizing functions resembling *augmented* Lagrangians.

Another noteworthy point is that in the minimization calculations for the dual proximal steps on $A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$, the matrix $\mathbf{M}$ does not appear in the argument of either f or g. By contrast, the primal proximal step $\mathbf{x}' \in (I + \lambda\partial[g \circ \mathbf{M}])^{-1}\overline{\mathbf{x}}$ involves minimizing a function including a term of the form $g(\mathbf{Mx})$. In practice, such a minimization might be inconvenient. Thus, in cases where $\mathbf{M}$ is not the identity matrix $\mathbf{I}$, dual splitting methods will generally be preferable to primal ones.

3.5.2 Background Material for Primal-Dual Symmetry

We are now almost ready to begin the last task of this chapter, which is to catalog the optimization algorithms produced by applying various splitting schemes to the primal splitting of (P), $A = \partial f$ and $B = \partial[g \circ \mathbf{M}]$, or the dual splitting of (P), $A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$. In the case $\mathbf{M} = \mathbf{I}$, we will sometimes see what appears to be an interesting coincidence: the primal and dual splittings will yield essentially the same algorithm. In this section, we give some preliminary material which should help to understand this phenomenon. All the material in this chapter concerning primal-dual symmetry is original.

The fundamental reason for primal-dual symmetry will turn out to be a fundamental relationship between the resolvent of a maximal monotone operator and the resolvent of its inverse. We now state the relationship in the simplest possible form.

Proposition 3.33. Let T be a maximal monotone operator on any Hilbert space $\mathcal{H}$. Then

$$J_T + J_{T^{-1}} = I \ .$$

Proof. Consider any $z \in \mathcal{H}$. Now, z may be expressed in exactly one way as $x + y$, where $(x, y) \in T$. We then have $J_T(z) = J_T(x + y) = x$, while $J_{T^{-1}}(z) = J_{T^{-1}}(x + y) = y$. Therefore, $J_T(z) + J_{T^{-1}}(z) = x + y = z$. As z was arbitrary, the result follows. ■

We now prove some slightly more complicated identities that are based on the same fundamental idea.

Proposition 3.34. Let A and B be any maximal monotone operators on a real Hilbert space $\mathcal{H}$, and let $A' = (-I)A^{-1}(-I)$. Let r be any fixed, positive real number. Then the following identities hold:

(i) $J_{(1/r)B^{-1}} = (\frac{1}{r}I)(I - J_{rB})(rI)$

(ii) $I - J_{(1/r)B^{-1}} = (\frac{1}{r}I)J_{rB}(rI)$

(iii) $2J_{(1/r)B^{-1}} - I = (-\frac{1}{r}I)(2J_{rB} - I)(rI)$

(iv) $J_{(1/r)A'} = (-\frac{1}{r}I)(I - J_{rA})(-rI)$

(v) $2J_{(1/r)A'} - I = (\frac{1}{r}I)(2J_{rA} - I)(-rI)$.

Proof. Consider any $z \in \mathcal{H}$, with unique representation $x + rb$, where $(x, b) \in B$, and $y + ra$, where $(y, a) \in A$. We then have

$$J_{(1/r)B^{-1}}(\tfrac{1}{r}z) = J_{(1/r)B^{-1}}(\tfrac{1}{r}x + b) = b = \tfrac{1}{r}[x + rb - x] = \tfrac{1}{r}[z - J_{rB}(z)] ,$$

hence $J_{(1/r)B^{-1}}(\frac{1}{r}I) = (\frac{1}{r}I)(I - J_{rB})$. Composing both sides of his identity with rI gives (i). Applying the left and right distributive laws of Lemma 3.3 to (i) and rearranging gives (ii). For (iii), we note that

$$(2J_{rB} - I)(z) = (2J_{rB} - I)(x + rb) = x - rb ,$$

while

$$(2J_{(1/r)B^{-1}} - I)(\tfrac{1}{r}z) = (2J_{(1/r)B^{-1}} - I)(\tfrac{1}{r}x + b) = -\tfrac{1}{r}x + b = (-\tfrac{1}{r})(x - rb) .$$

Therefore, $(2J_{(1/r)B^{-1}} - I)(\frac{1}{r}z) = (-\frac{1}{r})(2J_{rB} - I)(z)$. Since z is arbitrary, composing on the right with rI gives $2J_{(1/r)B^{-1}} - I = (-\frac{1}{r}I)(2J_{rB} - I)(rI)$.

For (iv), we have

$$J_{rA}(z) = J_{rA}(y + ra) = y$$

$$(I - J_{rA})(z) = y + ra - y = ra$$

$$J_{(1/r)A'}(-\tfrac{1}{r}z) = J_{(1/r)A'}(-\tfrac{1}{r}y - a) = -a = (-\tfrac{1}{r})(I - J_{rA})(z) \ .$$

Thus, $J_{(1/r)A'}(-\frac{1}{r}I) = (-\frac{1}{r}I)(I - J_{rA})$, and composing on the right with $-rI$, we obtain (iv).

The proof of (v) is similar: one has

$$(2J_{rA} - I)(z) = (2J_{rA} - I)(y + ra) = y - ra$$

$$(2J_{(1/r)A'} - I)(-\tfrac{1}{r}z) = (2J_{(1/r)A'} - I)(-\tfrac{1}{r}y - a) = \tfrac{1}{r}y - a \ ,$$

hence

$$2J_{(1/r)A'} - I = (\tfrac{1}{r}I)(2J_{rA} - I)(-rI) \ . \ \blacksquare$$

In the following sections, we will use the identities proved above to show that the Peaceman-Rachford and Douglas-Rachford methods display certain primal-dual symmetries.

3.5.3 *Double-Backward Optimization Methods*

We now consider optimization methods based on applying the double-backward splitting scheme to convex programs of the form (P). Because the double-backward method is known to converge only in the special case that A and B are normal cone operators of nonempty closed convex sets (see Section 3.4.1), such applications are scant.

Suppose one is given a finite-dimensional problem in the form (P),

$$\text{minimize}_{\mathbf{x}\in\mathbb{R}^n} \; f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \quad ,$$

and one takes $A = \partial f$ and $B = \partial[g \circ \mathbf{M}]$ in the double-backward method $\mathbf{x}^{k+1} = J_{\lambda_k B} J_{\lambda_k A} \mathbf{x}^k$. by Proposition 3.24(iii-iv), one then obtains the method

$$\begin{aligned} &\text{Start with any } \mathbf{x}^0 \in \mathbb{R}^n \\ &\mathbf{y}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}) + \frac{1}{2\lambda_k} \| \mathbf{x} - \mathbf{x}^k \|^2 \right\} \\ &\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ g(\mathbf{M}\mathbf{x}) + \frac{1}{2\lambda_k} \| \mathbf{x} - \mathbf{y}^{k+1} \|^2 \right\} \quad . \end{aligned}$$

In the case $\mathbf{M} = \mathbf{I}, f = \delta_C$, and $g = \delta_D$, where $C,D \subseteq \mathbb{R}^n$ are nonempty, closed, and convex, one obtains the method, starting with any $\mathbf{x}^0 \in \mathbb{R}^n$,

$$\mathbf{x}^{k+1} = P_D P_C(\mathbf{x}^k) \quad ,$$

which we know from Section 3.3 to converge to a point in the intersection of C and D, if one exists. One should also mention that this approach can be extended to find a point in the intersection of an arbitrary finite collection of nonempty closed convex sets $C_1,...,C_s$ by increasing the problem dimension to ns and setting

$$C = C_1 \times ... \times C_s \qquad D = \{ \mathbf{x} \in \mathbb{R}^{ns} \mid \mathbf{x}_1 = \mathbf{x}_2 = ... = \mathbf{x}_s \} \quad ,$$

where x_i denotes the vector comprising elements $n(i-1)+1$ through ni of $\mathbf{x}$. We then obtain the method, starting with any $\mathbf{x}^0 \in \mathbb{R}^n$,

$$\mathbf{x}^{k+1} = \frac{1}{s}\sum_{i=1}^{s} P_{C_i}(\mathbf{x}^k) \quad .$$

For further comments on this special case of the method, see, for example, Bertsekas and Tsitsiklis (1989). Apart from this sort of successive projection method, the applications of the double-backward method in optimization appear limited.

Now consider applying the double backward method to the dual splitting $A' = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B' = \partial g^*$, where $\mathbf{M}$ is assumed to have full column rank. Let us suppose that we are given some point $(\mathbf{p}^k, \mathbf{z}^k) \in \partial g^*$. Then the proximal step on A at $\mathbf{p}^k$, $\mathbf{q}^k = (I + \lambda_k \partial[f^* \circ (-\mathbf{M}^T)])^{-1}\mathbf{p}^k$, may be implemented, using Proposition 3.32(iv) with $\eta = \mathbf{z}^k$ and $\phi = \mathbf{0}$, by

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k + \lambda_k \mathbf{z}^k, \mathbf{M}\mathbf{x}\rangle + \frac{\lambda_k}{2}\|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2\}$$
$$\mathbf{q}^{k+1} = \mathbf{p}^k + \lambda_k \mathbf{M}\mathbf{x}^{k+1} \quad .$$

By Proposition 3.32(iii) with $\zeta = \mathbf{M}\mathbf{x}^{k+1}$ and $\theta = \mathbf{0}$, the proximal iteration on B at $\mathbf{q}^k$, $\mathbf{p}^{k+1} = (I + \lambda_k \partial g^*)^{-1}\mathbf{q}^k$, can be written

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{q}^{k+1} - \lambda_k \mathbf{M}\mathbf{x}^{k+1}, \mathbf{z}\rangle + \frac{\lambda_k}{2}\|\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}\|^2\}$$
$$\mathbf{p}^{k+1} = \mathbf{q}^{k+1} - \lambda_k \mathbf{z}^{k+1} \quad .$$

Combining these two calculations and eliminating the $\{\mathbf{q}^k\}$ gives the method

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \{f(\mathbf{x}) - \langle \mathbf{p}^k + \lambda_k \mathbf{z}^k, \mathbf{M}\mathbf{x}\rangle + \frac{\lambda_k}{2} \|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2\}$$
$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z}\rangle + \frac{\lambda_k}{2} \|\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}\|^2\}$$
$$\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda_k(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) \quad .$$

We mention this algorithm mainly for completeness; it does not appear to be particularly useful.

3.5.4 Forward-Backward Optimization Methods

We now consider the applications of the forward-backward splitting scheme to convex optimization, which are much more numerous than those of the double-backward scheme.

Consider first the primal splitting in which one takes $A = \partial f$ and $B = \partial[g \circ \mathbf{M}]$. Applying Proposition 3.24(i) and (iv), one obtains the method

$$\begin{aligned} &\text{Choose some } \mathbf{y}^k \in \partial f(\mathbf{x}^k) \qquad\qquad (\text{PFB}) \\ &\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ g(\mathbf{M}\mathbf{x}) + \frac{1}{2\lambda_k} \|\mathbf{x} - (\mathbf{x}^k - \lambda_k \mathbf{y}^k)\|^2 \right\} \quad . \end{aligned}$$

We call (PFB) the *primal forward-backward method.* If one takes $\mathbf{M} = \mathbf{I}$ and $g = \delta_C$, where $C$is a nonempty closed convex set, one obtains

$$\mathbf{x}^{k+1} \in P_C[\mathbf{x}^k - \lambda_k \partial f(\mathbf{x}^k)] \quad ,$$

the *subgradient projection* algorithm mentioned in Section 3.4.2. If one further assumes that f is differentiable throughout C, one gets the *gradient projection* method

$$\mathbf{x}^{k+1} = P_C[\mathbf{x}^k - \lambda_k \nabla f(\mathbf{x}^k)] \quad .$$

Results and references for both gradient and subgradient projection have already been given in Section 3.4.2.

As for the general convergence of the method (PFB), we can come up with a result which follows essentially directly from the Gabay/Tseng convergence theorem (Theorem 3.14). We need one lemma, whose proof may be found in Gol'shtein and Tret'yakov (1975).

Lemma 3.4. Let $h: \mathbb{R}^n \to \mathbb{R}$ be a convex, differentiable function. For any $L > 0$, the following two conditions are equivalent:

(i) $\| \nabla h(\mathbf{x}) - \nabla h(\mathbf{x}') \| \leq L \| \mathbf{x} - \mathbf{x}' \| \quad \forall\, \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$

(ii) $\langle \nabla h(\mathbf{x}) - \nabla h(\mathbf{x}'), \mathbf{x} - \mathbf{x}' \rangle \geq \frac{1}{L} \| \nabla h(\mathbf{x}) - \nabla h(\mathbf{x}') \|^2 \quad \forall\, \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$.

Proposition 3.35. Let the convex program (P),

$$\text{minimize}_{\ \mathbf{x} \in \mathbb{R}^n}\ f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \quad ,$$

and the sequence $\{\lambda_k\}_{k=0}^{\infty}$ obey all of the following conditions:

(i) $f: \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, and ∇f and obeys a Lipschitz condition

$$\| \nabla f(\mathbf{x}) - \nabla f(\mathbf{x}') \| \leq L \| \mathbf{x} - \mathbf{x}' \| \quad \forall\, \mathbf{x}, \mathbf{x}' \in \mathbb{R}^n \ ,$$

where $L > 0$.

(ii) $\mathbf{M}$ has full column rank.

(iii) (P) has a Kuhn-Tucker pair.

(iv) $0 < \inf_{k \geq 0} \lambda_k \leq \sup_{k \geq 0} \lambda_k < \frac{2}{L}$.

Then the sequence $\{\mathbf{x}^k\}$ generated by the method (PFB) converges to an optimal solution of (P).

Proof. By Lemma 3.4, $(\nabla f)^{-1}$ is strongly monotone operator of modulus $1/L$. Since (PFB) is an implementation of the forward-backward method on the operators $A = \nabla f$ and $B = \partial[g \circ \mathbf{M}]$, Theorem 3.14 implies that, given the conditions on λ_k, $\{\mathbf{x}^k\}$ converges to a zero of $A+B$, if one exists. As (P) has a Kuhn-Tucker pair, a zero of $A+B$ does indeed exist. Any zero of $A+B$ solves (P). ■

Essentially, the above proposition states that the gradient projection method retains its convergence properties when the projection operation is replaced with a proximal step on the subgradient of any convex function $g \circ \mathbf{M}$.

Above, we have taken a forward step on the operator ∂f, and a backward step on the operator $\partial[g \circ \mathbf{M}]$. In cases where $\mathbf{M} \neq \mathbf{I}$, it may prove more convenient to perform the forward step on $\partial[g \circ \mathbf{M}]$, and the backward step on ∂f. That is, one might wish to set $A = \partial[g \circ \mathbf{M}]$ and $B = \partial f$, and then apply the forward-backward method. Since the operators A and B play asymmetric roles in the forward-backward scheme, this approach results in a different algorithm. By proposition 3.24(ii-iii), this algorithm is

$$\begin{aligned} &\text{Choose some } \mathbf{y}^k \in \partial g(\mathbf{M}\mathbf{x}^k) \\ &\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}) + \frac{1}{2\lambda_k} \| \mathbf{x} - (\mathbf{x}^k - \lambda_k \mathbf{M}^T \mathbf{y}^k) \|^2 \right\} , \end{aligned} \qquad \text{(APFB)}$$

which we call the *alternate primal forward-backward method.* Assuming that g is everywhere differentiable gives the method

$$\mathbf{x}^{k+1} \in \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}) + \frac{1}{2\lambda_k} \| \mathbf{x} - (\mathbf{x}^k - \lambda_k \mathbf{M}^T \nabla g(\mathbf{M}\mathbf{x}^k)) \|^2 \right\} .$$

One may develop conditions for the convergence of this method similar to those of Proposition 3.35. In this case, one needs ∇g to be Lipschitzian,

$$\| \nabla g(\mathbf{z}) - \nabla g(\mathbf{z}') \| \leq L \| \mathbf{z} - \mathbf{z}' \| \quad \forall\, \mathbf{z}, \mathbf{z}' \in \mathbb{R}^m \ ,$$

and the stepsize condition is

$$0 < \inf_{k \geq 0} \lambda_k \leq \sup_{k \geq 0} \lambda_k < \frac{2}{L\rho(\mathbf{M}^T\mathbf{M})} \ ,$$

where $\rho(\mathbf{M}^T\mathbf{M})$ denotes the spectral radius of $\mathbf{M}^T\mathbf{M}$.

We now consider dual applications of the forward-backward method, to wit, letting $A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$. Appealing to Proposition 3.32(ii) with $\phi = \mathbf{0}$, we obtain that the forward step $\mathbf{q}^{k+1} = (I - \lambda_k A)\mathbf{p}^k$ may be performed via

$$\begin{aligned} &\text{Choose } \mathbf{x}^{k+1} \in \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle\} \\ &\mathbf{q}^{k+1} = \mathbf{p}^k + \lambda_k \mathbf{M}\mathbf{x}^{k+1} \ . \end{aligned}$$

Then, using Proposition 3.32(iii) with $\zeta = \mathbf{M}\mathbf{x}^{k+1}$ and $\theta = \mathbf{0}$, one can implement the proximal step $\mathbf{p}^{k+1} = (I + \lambda_k B)^{-1}\mathbf{q}^{k+1}$ by

$$\begin{aligned} &\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{q}^{k+1} - \lambda_k \mathbf{M}\mathbf{x}^{k+1}, \mathbf{z} \rangle + \frac{\lambda_k}{2} \| \mathbf{M}\mathbf{x}^{k+1} - \mathbf{z} \|^2\} \\ &\mathbf{p}^{k+1} = \mathbf{q}^{k+1} - \lambda_k \mathbf{z}^{k+1} \ . \end{aligned}$$

Combining these two steps and eliminating the $\{\mathbf{q}^k\}$, one obtains the *dual forward-backward method*,

$$\begin{aligned} &\text{Choose } \mathbf{x}^{k+1} \in \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle\} \\ &\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z} \rangle + \frac{\lambda_k}{2} \| \mathbf{M}\mathbf{x}^{k+1} - \mathbf{z} \|^2\} \qquad \text{(DFB)} \\ &\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda_k(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) \ . \end{aligned}$$

We can now apply Theorem 3.14, proving a variation on a result of Tseng (1988):

Proposition 3.36. Suppose we have a dual normal convex program in the form (P),

$$\text{minimize}_{\mathbf{x}\in\mathbb{R}^n}\ f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \quad ,$$

that possesses a Kuhn-Tucker pair. Suppose also that f is strongly monotone with modulus $\alpha > 0$, and $\mathbf{M}$ has full column rank. Then for any initial $\mathbf{p}^0 \in \mathbb{R}^m$, the sequences $\{\mathbf{p}^k\}$ and $\{\mathbf{x}^k\}$ evolved by the method (DFB) under the stepsize restriction

$$0 < \inf_{k\geq 0} \lambda_k \leq \sup_{k\geq 0} \lambda_k < \frac{2\alpha}{\rho(\mathbf{M}^T\mathbf{M})} \quad ,$$

converge to a solution of (D) and the unique solution of (P), respectively (again, $\rho(\mathbf{M}^T\mathbf{M})$ denotes the spectral radius of $\mathbf{M}^T\mathbf{M}$).

Proof. The strong monotonicity of f implies the uniqueness of the solution to (P), which exists by the assumption that (P) has a Kuhn-Tucker pair. It also follows from strong monotonicity that im $\partial f = \mathbb{R}^n$ (using Theorem 3.5), and so

$$f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x}\rangle = f(\mathbf{x}) + \langle \mathbf{M}^T\mathbf{p}^k, \mathbf{x}\rangle$$

has a unique minimum for any $\mathbf{p}^k$. As $\mathbf{M}$ has full row rank, the second minimization in (DFB) must also always have a solution, and so (DFB) can be executed indefinitely from any starting point. We first show the convergence of $\{\mathbf{p}^k\}$ to some solution of (D), which must also exist by the Kuhn-Tucker assumption. In view of Theorem 3.14, we need only show that $A^{-1} = (\partial[f^* \circ (-\mathbf{M}^T)])^{-1}$ is strongly monotone with modulus $\alpha/\rho(\mathbf{M}^T\mathbf{M})$. Since (P) is dual normal, $\partial[f^* \circ (-\mathbf{M}^T)] = -\mathbf{M} \circ \partial f^* \circ (-\mathbf{M}^T)$. By assumption, $(\partial f^*)^{-1} = \partial f$ is strongly monotone with modulus α. We have

$$A = -\mathbf{M} \circ \partial f^* \circ (-\mathbf{M}^T) = \{(\mathbf{w}, \mathbf{z}) \mid \exists\ (\mathbf{x}, \mathbf{y}) \in \partial f:\ -\mathbf{M}^T\mathbf{w} = \mathbf{y}, \mathbf{z} = -\mathbf{M}\mathbf{x}\} \ .$$

Choose any $(\mathbf{w}, \mathbf{z}), (\mathbf{w}', \mathbf{z}') \in A$. Let $(\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}') \in \partial f$ be such that $-\mathbf{M}^T\mathbf{w} = \mathbf{y}$, $\mathbf{z} = -\mathbf{M}\mathbf{x}$, $-\mathbf{M}^T\mathbf{w}' = \mathbf{y}'$, and $\mathbf{z}' = -\mathbf{M}\mathbf{x}'$. Then

$$
\begin{aligned}
\langle \mathbf{w} - \mathbf{w}', \mathbf{z} - \mathbf{z}' \rangle &= \langle \mathbf{w} - \mathbf{w}', -\mathbf{M}\mathbf{x} + \mathbf{M}\mathbf{x}' \rangle \\
&= \langle -\mathbf{M}^T\mathbf{w} + \mathbf{M}^T\mathbf{w}', \mathbf{x} - \mathbf{x}' \rangle \\
&= \langle \mathbf{y} - \mathbf{y}', \mathbf{x} - \mathbf{x}' \rangle \\
&\geq \alpha \| \mathbf{x} - \mathbf{x}' \|^2 \\
&\geq \frac{\alpha}{\rho(\mathbf{M}^T\mathbf{M})} \| \mathbf{M}\mathbf{x} - \mathbf{M}\mathbf{x}' \|^2 \\
&= \frac{\alpha}{\rho(\mathbf{M}^T\mathbf{M})} \| \mathbf{z} - \mathbf{z}' \|^2 \ ,
\end{aligned}
$$

where the second inequality follows from $\| \mathbf{M}\mathbf{x} - \mathbf{M}\mathbf{x}' \|^2 \leq \rho(\mathbf{M}^T\mathbf{M}) \| \mathbf{x} - \mathbf{x}' \|^2$ for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$. Thus, A^{-1} has the desired property and $\{\mathbf{p}^k\}$ converges to a solution $\mathbf{p}^*$ of (D). We must now establish the convergence of $\{\mathbf{x}^k\}$. Since $(\partial f^*)^{-1} = \partial f$ is strongly monotone, ∂f^* is single-valued and Lipschitz continuous. From the construction of the forward step, we have $(\mathbf{x}^{k+1}, -\mathbf{M}^T\mathbf{p}^k) \in \partial f$, hence $(-\mathbf{M}^T\mathbf{p}^k, \mathbf{x}^{k+1}) \in \partial f^*$, for all $k \geq 0$. From the continuity of ∂f^*, $\mathbf{x}^{k+1}$ converges to the limit $\mathbf{x}^* = \partial f^*(-\mathbf{M}^T\mathbf{p}^*)$. The convergence of the $\mathbf{p}^k$, the identity $\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda_k(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1})$ for all $k \geq 0$, and $\{\lambda_k\}$ being bounded away from 0 together imply that $\|\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}\| \to 0$, hence $\{\mathbf{z}^k\}$ converges to $\mathbf{M}\mathbf{x}^*$. Furthermore, Proposition 3.32(iii) gives that $(\mathbf{p}^{k+1}, \mathbf{z}^{k+1}) \in \partial g^*$ for all k, and so, by Proposition 3.2, $\mathbf{M}\mathbf{x}^* \in \partial g^*(\mathbf{p}^*)$. Thus we have $(\mathbf{x}^*, -\mathbf{M}^T\mathbf{p}^*) \in \partial f$, $(\mathbf{p}^*, \mathbf{M}\mathbf{x}^*) \in \partial g$, and so $\mathbf{x}^* = \lim_{k \to 0} \mathbf{x}^k$ is a solution of (P). ■

One special case of the dual forward-backward method is an algorithm recently proposed by Han and Lou (1988) for solving problems of the form

$$
\begin{aligned}
&\min \ h(\mathbf{x}) \\
&\text{ST} \ \ \mathbf{x} \in C_1 \cap \ldots \cap C_s \ ,
\end{aligned}
\qquad \text{(HLP)}
$$

where h is closed, proper, and strongly convex, and $C_1,\ldots, C_s \subseteq \mathbb{R}^n$ are closed convex sets having nonempty intersection. One can convert this problem into the from (P) as follows: let $\mathbf{M}$ be the $ns \times n$ matrix

$$\mathbf{M} = \left[\begin{array}{c} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{array}\right] \Bigg\} s \text{ times} \quad ,$$

let $f = h$, and let g: $(\mathbb{R}^n)^s \rightarrow (-\infty, \infty]$ be the function $\delta_{C_1 \times \ldots \times C_s}$, that is, $g(\mathbf{z}) = 0$ if $\mathbf{z} \in C_1 \times \ldots \times C_s$, otherwise $g(\mathbf{z}) = \infty$. It should be clear that for this choice of f, g, and $\mathbf{M}$, (P) is equivalent to (HLP). Tseng (1988) has shown that applying the algorithm (DFB) in this case yields the method proposed by Han and Lou.

3.5.5 Peaceman-Rachford Optimization Methods

This section will present optimization methods obtained by applying the Peaceman-Rachford scheme to the primal and dual splittings of convex programs in the form (P). In keeping with the convergence theory presented in Section 3.4.3, we will concentrate on the case in which the stepsizes λ_k are held constant.

Consider first the case of the primal splitting $A = \partial f$ and $B = \partial[g \circ \mathbf{M}]$. To see what kind of algorithm this choice of A and B implies, we restate the "tight" Peaceman-Rachford method of Section 3.4.3, making a slight change of notation to avoid conflict with the notation we have used for (P) and (D):

(1) Given $(\mathbf{w}^k, \mathbf{b}^k) \in B$, find the unique $(\mathbf{y}^{k+1}, \mathbf{a}^{k+1}) \in A$ such that
$\mathbf{y}^{k+1} + \lambda_k \mathbf{a}^{k+1} = \mathbf{w}^k - \lambda_k \mathbf{b}^k$

(2) Find the unique $(\mathbf{w}^{k+1}, \mathbf{b}^{k+1}) \in B$ such that $\mathbf{w}^{k+1} + \lambda_k \mathbf{b}^{k+1} = \mathbf{y}^{k+1} - \lambda_k \mathbf{a}^{k+1}$.

The choice of A and B and Proposition 3.24(iii-iv) imply that these steps may be implemented as follows:

$$\begin{aligned}
\mathbf{y}^{k+1} &= \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}) + \frac{1}{2\lambda_k} \| \mathbf{x} - (\mathbf{w}^k - \lambda \mathbf{b}^k) \|^2 \right\} \\
\mathbf{a}^{k+1} &= \frac{1}{\lambda_k}(\mathbf{w}^k - \lambda_k \mathbf{b}^k - \mathbf{y}^{k+1}) \\
\mathbf{w}^{k+1} &= \arg\min_{\mathbf{x}} \left\{ g(\mathbf{M}\mathbf{x}) + \frac{1}{2\lambda_k} \| \mathbf{x} - (\mathbf{y}^{k+1} - \lambda_k \mathbf{a}^{k+1}) \|^2 \right\} \\
\mathbf{b}^{k+1} &= \frac{1}{\lambda_k}(\mathbf{y}^{k+1} - \lambda_k \mathbf{a}^{k+1} - \mathbf{w}^{k+1}) \quad .
\end{aligned}$$

Eliminating the $\{\mathbf{a}^k\}$ and simplifying, one obtains the *primal Peaceman-Rachford algorithm*

$$\begin{aligned}
\mathbf{y}^{k+1} &= \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}) + \frac{1}{2\lambda_k} \| \mathbf{x} - (\mathbf{w}^k - \lambda \mathbf{b}^k) \|^2 \right\} \\
\mathbf{w}^{k+1} &= \arg\min_{\mathbf{x}} \left\{ g(\mathbf{M}\mathbf{x}) + \frac{1}{2\lambda_k} \| \mathbf{x} - (2\mathbf{y}^{k+1} - \mathbf{w}^k + \lambda_k \mathbf{b}^k) \|^2 \right\} \qquad \text{(PPR)} \\
\mathbf{b}^{k+1} &= \mathbf{b}^k + \frac{1}{\lambda_k}(2\mathbf{y}^{k+1} - \mathbf{w}^{k+1} - \mathbf{w}^k) \quad .
\end{aligned}$$

We can start this algorithm from any $(\mathbf{w}^0, \mathbf{b}^0) \in \mathbb{R}^n \times \mathbb{R}^n$; there is no need to have $(\mathbf{w}^0, \mathbf{b}^0) \in B$, since, by construction, we will automatically have $(\mathbf{w}^k, \mathbf{b}^k) \in B$ for all $k \geq 1$, and the algorithm will automatically "get on track".

Corollary 3.17.1 provides sufficient conditions for the (PPR) to converge:

Proposition 3.37. Consider a primal splittable, solvable convex program in the form (P). If all the stepsizes λ_k are held constant at some fixed $\lambda > 0$ and *either* of the following conditions hold

(i) f is strongly convex and everywhere finite and differentiable,

(ii) g is finite and differentiable throughout im $\mathbf{M}$, and strongly convex when restricted to im $\mathbf{M}$,

then $\{\mathbf{w}^k\}$ generated by the method (PPR), for any starting point $(\mathbf{w}^0, \mathbf{b}^0) \in \mathbb{R}^n \times \mathbb{R}^n$, converges to the unique solution of (P).

Proof. Condition (i) implies that ∂f is single-valued and strongly monotone, while (ii) implies that $\partial[g \circ \mathbf{M}]$ is single-valued and strongly monotone. Thus, either (i) or (ii) suffices to meet the conditions of Corollary 3.17.1, and the sequence $\{\mathbf{t}^k\} = \{\mathbf{w}^k + \lambda \mathbf{b}^k\}$ converges to some element $\mathbf{t}^*$ of $T_\lambda^* = \{\, \mathbf{w} + \lambda \mathbf{b} \mid (\mathbf{w}, \mathbf{b}) \in B, (\mathbf{w}, -\mathbf{b}) \in A \}$, which is nonempty by the assumed splittability and solvability of (P). Let $(\mathbf{w}^*, \mathbf{b}^*) \in B$ be the unique pair such that $\mathbf{w}^* + \lambda \mathbf{b}^* = \mathbf{t}^*$. By the continuity of the resolvent $J_{\lambda B}$,

$$\lim_{k \to \infty} \mathbf{w}^k = \lim_{k \to \infty} J_{\lambda B}(\mathbf{t}^k) = J_{\lambda B}(\mathbf{t}^*) = \mathbf{w}^* .$$

By the form of T_λ^*, $\mathbf{w}^*$ is a zero of $A+B$, and must solve (P). The solution is necessarily unique because (i) and (ii) each imply that $f + g \circ \mathbf{M}$ is strongly convex. ■

We now consider applying the Peaceman-Rachford scheme to dual splittings of the form $\partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$. We restate the tight Peaceman-Rachford method in notation suitable for a dual algorithm:

(1') Given $(\mathbf{p}^k, \mathbf{z}^k) \in B$, find the unique $(\mathbf{q}^{k+1}, \mathbf{a}^{k+1}) \in A$ such that $\mathbf{q}^{k+1} + \lambda_k \mathbf{a}^{k+1} = \mathbf{p}^k - \lambda_k \mathbf{z}^k$

(2') Find the unique $(\mathbf{p}^{k+1}, \mathbf{z}^{k+1}) \in B$ such that $\mathbf{p}^{k+1} + \lambda_k \mathbf{z}^{k+1} = \mathbf{q}^{k+1} - \lambda_k \mathbf{a}^{k+1}$.

Proposition 3.32(iv) with $\bar{\mathbf{p}} = \mathbf{p}^k - \lambda_k \mathbf{z}^k$, $\eta = \lambda_k \mathbf{z}^k$, and $\phi = \mathbf{0}$ implies that (1') can be implemented via

$$\begin{aligned}
\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle + \frac{\lambda_k}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z}^k \|^2 \} \\
\mathbf{q}^{k+1} &= \mathbf{p}^k + \lambda_k (\mathbf{M}\mathbf{x}^{k+1} - \lambda_k \mathbf{z}^k) \\
\mathbf{a}^{k+1} &= -\lambda_k \mathbf{M}\mathbf{x}^{k+1} \quad .
\end{aligned}$$

Then, from Proposition 3.32(iii) with

$$\bar{\mathbf{p}} = \mathbf{q}^{k+1} - \lambda_k \mathbf{a}^{k+1} = \mathbf{p}^k + \lambda_k (2\mathbf{M}\mathbf{x}^{k+1} - \lambda_k \mathbf{z}^k) \quad ,$$

$\zeta = \mathbf{M}\mathbf{x}^{k+1}$, and $\theta = \mathbf{0}$, (2') may be performed via

$$\begin{aligned}
\mathbf{z}^{k+1} &= \arg\min_{\mathbf{z}} \{ g(\mathbf{z}) - \langle \mathbf{p}^k + \lambda_k (\mathbf{M}\mathbf{x}^{k+1} - \lambda_k \mathbf{z}^k), \mathbf{z} \rangle + \frac{\lambda_k}{2} \| \mathbf{z} - \mathbf{M}\mathbf{x}^{k+1} \|^2 \} \\
\mathbf{p}^{k+1} &= \mathbf{p}^k + \lambda_k (2\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^k - \mathbf{z}^{k+1}) \quad .
\end{aligned}$$

Combining these two calculations and simplifying yields the *dual Peaceman-Rachford algorithm*

$$\begin{aligned}
\mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle + \frac{\lambda_k}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z}^k \|^2 \} \\
\mathbf{q}^{k+1} &= \mathbf{p}^k + \lambda_k (\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^k) \qquad \text{(DPR)} \\
\mathbf{z}^{k+1} &= \arg\min_{\mathbf{z}} \{ g(\mathbf{z}) - \langle \mathbf{q}^{k+1}, \mathbf{z} \rangle + \frac{\lambda_k}{2} \| \mathbf{M}\mathbf{x}^{k+1} - \mathbf{z} \|^2 \} \\
\mathbf{p}^{k+1} &= \mathbf{q}^{k+1} + \lambda_k (\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) \quad .
\end{aligned}$$

This algorithm was first proposed in Gabay (1983), and was also analyzed in Glowinski and Le Tallec (1987). Recalling the reformulation of (P) as (P'),

$$\begin{aligned}
&\text{minimize} \quad f(\mathbf{x}) + g(\mathbf{z}) \\
&\text{subject to} \quad \mathbf{M}\mathbf{x} = \mathbf{z} \quad ,
\end{aligned}$$

the dual Peaceman-Rachford algorithm has the following interpretation:

(I) Minimize the augmented Lagrangian for (P') with respect to $\mathbf{x}$ only.

(II) Update the Lagrange multipliers in the same manner as in the conventional method of multipliers.

(III) Minimize the augmented Lagrangian for (P') with respect to $\mathbf{z}$ only.

(IV) Again, update the Lagrange multipliers.

Thus, the method compares interestingly with the usual method of multipliers for (P'), in which one minimizes the augmented Lagrangian with respect to both $\mathbf{x}$ and $\mathbf{z}$ simultaneously, and then does a multiplier update.

Corollary 3.17.1 also furnishes conditions under which (DPR) converges.

Proposition 3.38. Fix $\lambda > 0$ and suppose that $\lambda_k = \lambda$ for all $k \geq 0$. Consider any convex program in the form (P),

$$\text{minimize}_{\ \mathbf{x} \in \mathbb{R}^n}\ f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \quad ,$$

possessing a Kuhn-Tucker pair. If *either* of the following conditions hold,

(i) f is strictly convex, ∂f obeys a Lipschitz condition (and hence must be single-valued), and $\mathbf{M}$ is square and invertible,

(ii) g is strictly convex, ∂g obeys a Lipschitz condition (and hence is single-valued), and $\mathbf{M}$ has full column rank,

then for any starting point $(\mathbf{p}^0, \mathbf{z}^0) \in \mathbb{R}^m \times \mathbb{R}^m$, the sequences $\{\mathbf{x}^k\}$ and $\{\mathbf{p}^k\}$ generated by the recursion (DPR) converge to solutions of (P) and (D) respectively.

Proof. First consider condition (ii). Lemma 3.4 implies that ∂g^* is strongly monotone,

and the strict convexity of g implies that ∂g^* is single-valued. Thus $B = \partial g^*$ satisfies the conditions of Corollary 3.17.1. Now consider condition (i). By similar reasoning, f^* is strongly convex and differentiable. Therefore $f^* \circ (-\mathbf{M}^T)$ is differentiable, and since the null space of $\mathbf{M}^T$ must be $\{\mathbf{0}\}$, $f^* \circ (-\mathbf{M}^T)$ is also strongly convex. Therefore $\partial[f^* \circ (-\mathbf{M}^T)]$ is strongly monotone and single-valued, meeting the conditions of Corollary 3.17.1.

So, if either (i) or (ii) holds, Corollary 3.17.1 implies that $\{\mathbf{t}^k\} = \{\mathbf{p}^k + \lambda \mathbf{z}^k\}$ converges to some element $\mathbf{t}^*$ of $T_\lambda^* = \{\ \mathbf{p} + \lambda \mathbf{z}\ \mid\ (\mathbf{p}, \mathbf{z}) \in B,\ (\mathbf{p}, -\mathbf{z}) \in A\}$, if one exists. T_λ^* is indeed nonempty by the assumed existence of a Kuhn-Tucker pair. Applying the continuous operator $J_{\lambda B}$ to $\{\mathbf{t}^k\}$, we obtain that $\mathbf{p}^k \to \mathbf{p}^*$ and $\mathbf{z}^k \to \mathbf{z}^*$, where $(\mathbf{p}^*, \mathbf{z}^*) \in B$ is the unique pair such that $\mathbf{p}^* + \lambda \mathbf{z}^* = \mathbf{t}^*$. From (DPR), we have

$$\mathbf{p}^{k+1} \ = \ \mathbf{p}^k + \lambda_k(2\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1} - \mathbf{z}^k) \ ,$$

or,

$$\mathbf{M}\mathbf{x}^{k+1} \ = \ \tfrac{1}{2}(\mathbf{z}^{k+1} + \mathbf{z}^k - \tfrac{1}{\lambda}(\mathbf{p}^{k+1} - \mathbf{p}^k)) \quad .$$

Taking limits and using the convergence of $\{\mathbf{p}^k\}$ and $\{\mathbf{z}^k\}$, one obtains that $\mathbf{M}\mathbf{x}^{k+1} \to \mathbf{z}^*$. Both (i) and (ii) imply that $\mathbf{M}$ has full column rank, so for $\{\mathbf{M}\mathbf{x}^k\}$ to be convergent, $\{\mathbf{x}^k\}$ must be convergent. Letting $\mathbf{x}^*$ be the limit of $\{\mathbf{x}^k\}$, we then have $\mathbf{M}\mathbf{x}^* = \mathbf{z}^*$. From Proposition 3.32(iv),

$$(\mathbf{x}^{k+1}, -\mathbf{M}^T\mathbf{q}^{k+1}) \ = \ (\mathbf{x}^{k+1}, -\mathbf{M}^T[\mathbf{p}^k + \lambda_k(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^k)]) \ \in \ \partial f$$

for all k. Taking limits and using Proposition 3.2, $(\mathbf{x}^*, -\mathbf{M}^T\mathbf{p}^*) \in \partial f$. Since $(\mathbf{p}^*, \mathbf{z}^*) = (\mathbf{p}^*, \mathbf{M}\mathbf{x}^*) \in B = \partial g^*$, we have $(\mathbf{x}^*, -\mathbf{M}^T\mathbf{p}^*) \in \partial f$ and $(\mathbf{M}\mathbf{x}^*, \mathbf{p}^*) \in \partial g$, and so by the Kuhn-Tucker theorem, $\mathbf{x}^*$ is optimal for (P) and $\mathbf{p}^*$ is optimal for (D). ■

Finally, we show that the Peaceman-Rachford scheme conforms to a certain primal-dual symmetry. First, we state a slightly more general result that serves as an underpinning for this observation.

Lemma 3.5. Let r be any positive real number and let A and B be maximal monotone operators on any Hilbert space $\mathcal{H}$. Let $A' = (-I)A^{-1}(-I)$ and $B' = B^{-1}$. If one then defines

$$\begin{aligned} H &= (2J_{rA} - I)(2J_{rB} - I) \\ H' &= (2J_{(1/r)A'} - I)(2J_{(1/r)B'} - I) \quad , \end{aligned}$$

then $H' = (\frac{1}{r}I)H(rI)$.

Proof. Recall the identities proved in Proposition 3.34, which state that

$$\begin{aligned} 2J_{(1/r)B^{-1}} - I &= (-\tfrac{1}{r}I)(2J_{rB} - I)(rI) \\ 2J_{(1/r)A'} - I &= (\tfrac{1}{r}I)(2J_{rA} - I)(-rI) \quad . \end{aligned}$$

Then

$$\begin{aligned} H' &= (\tfrac{1}{r}I)(2J_{rA} - I)(-rI)(-\tfrac{1}{r}I)(2J_{rB} - I)(rI) \\ &= (\tfrac{1}{r}I)(2J_{rA} - I)(2J_{rB} - I)(rI) \\ &= (\tfrac{1}{r}I)H(rI) \quad . \quad \blacksquare \end{aligned}$$

Proposition 3.39. The tight Peaceman-Rachford splitting scheme is primal-dual symmetric in the following sense: let $\mathbf{M} = \mathbf{I}$ and suppose the primal Peaceman-Rachford algorithm (PPR) generate the sequence $\{(\mathbf{w}^k, \mathbf{b}^k)\}$, starting from some $(\mathbf{w}^0, \mathbf{b}^0) \in \mathbb{R}^n \times \mathbb{R}^n$, while the dual Peaceman-Rachford algorithm (DPR) generates the sequence $\{(\mathbf{p}^k, \mathbf{z}^k)\}$, starting from some $(\mathbf{p}^0, \mathbf{z}^0) \in \mathbb{R}^n \times \mathbb{R}^n$. Let $\{\lambda_k\}$ denote the stepsizes used in the primal, and $\{\lambda_k'\}$ the stepsizes used in the dual. Then if $\lambda_k = 1/\lambda_k'$ for all $k \geq 0$, $\mathbf{w}^0 = \mathbf{z}^0$, and $\mathbf{b}^0 = \mathbf{p}^0$, one has $\mathbf{w}^k = \mathbf{z}^k$ and $\mathbf{b}^k = \mathbf{p}^k$ for all $k \geq 0$.

Proof. Considering that $\mathbf{M} = \mathbf{I}$, the primal Peaceman-Rachford algorithm uses the tight Peaceman-Rachford splitting scheme as applied to the operators $A = \partial f$ and $B = \partial g$, while the dual Peaceman-Rachford algorithm applies it to

$$A' = \partial[f^* \circ (-\mathbf{I})] = (-\mathbf{I})\partial f^*(-\mathbf{I}) = (-\mathbf{I})A^{-1}(-\mathbf{I})$$

and $B' = \partial g^* = B^{-1}$. We now use induction to prove $\mathbf{w}^k = \mathbf{z}^k$ and $\mathbf{b}^k = \mathbf{p}^k$ for all $k \geq 0$. The base case $\mathbf{w}^0 = \mathbf{z}^0$, and $\mathbf{b}^0 = \mathbf{p}^0$ is true by hypothesis. Now fix any k and assume that $\mathbf{w}^k = \mathbf{z}^k$ and $\mathbf{b}^k = \mathbf{p}^k$. Set $r = \lambda_k$ and define H and H' as in Lemma 3.5:

$$\begin{aligned} H &= (2J_{rA} - I)(2J_{rB} - I) \\ H' &= (2J_{(1/r)A'} - I)(2J_{(1/r)B'} - I) \quad . \end{aligned}$$

The procedure for obtaining $(\mathbf{w}^{k+1}, \mathbf{b}^{k+1})$ may be expressed as follows:

$$\mathbf{t} = (2J_{rA} - I)(2J_{rB} - I)(\mathbf{w}^k + r\mathbf{b}^k) = H(\mathbf{w}^k + r\mathbf{b}^k)$$

Find the unique $(\mathbf{w}^{k+1}, \mathbf{b}^{k+1}) \in B$ such that $\mathbf{w}^{k+1} + r\mathbf{b}^{k+1} = \mathbf{t}$.

The procedure for obtaining $(\mathbf{p}^{k+1}, \mathbf{z}^{k+1})$ from $(\mathbf{p}^k, \mathbf{z}^k)$ in the dual method may be written as follows:

$$\mathbf{u} = (2J_{(1/r)A'} - I)(2J_{(1/r)B'} - I)(\mathbf{p}^k + \tfrac{1}{r}\mathbf{z}^k) = H'(\mathbf{p}^k + \tfrac{1}{r}\mathbf{z}^k)$$

Find the unique $(\mathbf{p}^{k+1}, \mathbf{z}^{k+1}) \in B'$ such that $\mathbf{p}^{k+1} + \frac{1}{r}\mathbf{z}^{k+1} = \mathbf{u}$.

Using the identity $H' = (\frac{1}{r}I)H(rI)$ provided by the lemma,

$$\begin{aligned} \mathbf{u} &= H'(\mathbf{p}^k + \tfrac{1}{r}\mathbf{z}^k) \\ &= (\tfrac{1}{r}I)H(rI)(\mathbf{p}^k + \tfrac{1}{r}\mathbf{z}^k) \\ &= (\tfrac{1}{r}I)H(\mathbf{z}^k + r\mathbf{p}^k) \end{aligned}$$

$$= (\tfrac{1}{r} I) H(\mathbf{w}^k + r\mathbf{b}^k)$$
$$= \tfrac{1}{r}\mathbf{t} \ .$$

We then have

$$\begin{aligned}
\mathbf{p}^{k+1} &= J_{(1/r)B'}(\mathbf{u}) \\
&= J_{(1/r)B'}(\tfrac{1}{r}\mathbf{t}) \\
&= J_{(1/r)B'}(\tfrac{1}{r}(\mathbf{w}^{k+1} + r\mathbf{b}^{k+1})) \\
&= J_{(1/r)B'}(\mathbf{b}^{k+1} + \tfrac{1}{r}\mathbf{w}^{k+1}) \\
&= \mathbf{b}^{k+1} \ ,
\end{aligned}$$

since $(\mathbf{w}^{k+1}, \mathbf{b}^{k+1}) \in B$ implies $(\mathbf{b}^{k+1}, \mathbf{w}^{k+1}) \in B^{-1} = B'$. Similarly, $\mathbf{z}^{k+1} = \mathbf{w}^{k+1}$. This completes the induction step, and the claim is established. ■

The same form of primal-dual symmetry may be deduced by "completing the square" in the iteration (DPR) to remove the linear terms from the two minimization arguments, and then noticing that the resulting iteration is identical to (PPR), except that the roles of $\{\lambda_k\}$ and $\{\frac{1}{\lambda_k}\}$ are reversed. The above proof may seem less direct, but reveals that this apparent coincidence is a result of the symmetry between the operators H and H'. This symmetry, in turn, stems essentially from the symmetry between the resolvents of rT and $\frac{1}{r}T^{-1}$, where $r > 0$ and T is any maximal monotone operator.

3.5.6 Douglas-Rachford Optimization Methods

The final topic of this chapter will be primal and dual optimization methods based on the Douglas-Rachford splitting scheme. Because of the generality with which the Douglas-Rachford scheme converges, these optimization methods will be more robust than those presented in the preceding sections.

First, consider the primal splitting $A = \partial f$ and $B = \partial[g \circ \mathbf{M}]$. We restate the tight Douglas-Rachford splitting method in a slightly altered notation:

(a'') Given $(\mathbf{w}^k, \mathbf{b}^k) \in B$, find the unique $(\mathbf{y}^{k+1}, \mathbf{a}^{k+1}) \in A$ such that $\mathbf{y}^{k+1} + \lambda_k \mathbf{a}^{k+1} = \mathbf{w}^k - \lambda_k \mathbf{b}^k$

(b'') Find the unique $(\mathbf{w}^{k+1}, \mathbf{b}^{k+1}) \in B$ such that $\mathbf{w}^{k+1} + \lambda_k \mathbf{b}^{k+1} = \mathbf{y}^{k+1} + \lambda_k \mathbf{b}^k$.

From Proposition 3.24(iii), (a'') may be implemented as

$$\mathbf{y}^{k+1} = \arg\min_{\mathbf{y}} \left\{ f(\mathbf{y}) + \frac{1}{2\lambda_k} \| \mathbf{y} - (\mathbf{w}^k - \lambda_k \mathbf{b}^k) \|^2 \right\}$$
$$\mathbf{a}^{k+1} = \frac{1}{\lambda_k}(\mathbf{w}^k - \lambda_k \mathbf{b}^k - \mathbf{y}^{k+1}) = -\mathbf{b}^k + \frac{1}{\lambda_k}(\mathbf{w}^k - \mathbf{y}^{k+1}) \quad .$$

Meanwhile, Proposition 3.24(iv) implies that (b'') may be written

$$\mathbf{w}^{k+1} = \arg\min_{\mathbf{x}} \left\{ g(\mathbf{M}\mathbf{x}) + \frac{1}{2\lambda_k} \| \mathbf{x} - (\mathbf{y}^{k+1} + \lambda_k \mathbf{b}^k) \|^2 \right\}$$
$$\mathbf{b}^{k+1} = \frac{1}{\lambda_k}(\mathbf{y}^{k+1} + \lambda_k \mathbf{b}^k - \mathbf{w}^{k+1}) = \mathbf{b}^k + \frac{1}{\lambda}(\mathbf{y}^{k+1} - \mathbf{w}^{k+1}) \quad .$$

Combining these two calculations and simplifying yields the *primal Douglas-Rachford* algorithm,

$$\begin{aligned} \mathbf{y}^{k+1} &= \arg\min_{\mathbf{y}} \left\{ f(\mathbf{y}) + \frac{1}{2\lambda_k} \| \mathbf{y} - (\mathbf{w}^k - \lambda_k \mathbf{b}^k) \|^2 \right\} \\ \mathbf{w}^{k+1} &= \arg\min_{\mathbf{x}} \left\{ g(\mathbf{M}\mathbf{x}) + \frac{1}{2\lambda_k} \| \mathbf{x} - (\mathbf{y}^{k+1} + \lambda_k \mathbf{b}^k) \|^2 \right\} \\ \mathbf{b}^{k+1} &= \mathbf{b}^k + \frac{1}{\lambda}(\mathbf{y}^{k+1} - \mathbf{w}^{k+1}) \quad . \end{aligned} \tag{PDR}$$

The convergence properties of this algorithm are quite general:

Proposition 3.40. Consider a primal splittable, solvable convex program in the form (P),

$$\text{minimize}_{\mathbf{x}\in\mathbb{R}^n}\ f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \quad .$$

Fix some $\lambda > 0$, and let $\lambda_k = \lambda$ for all $k \geq 0$. For any starting point $(\mathbf{w}^0, \mathbf{b}^0) \in \mathbb{R}^n \times \mathbb{R}^n$, the sequence $\{\mathbf{w}^k\}$ generated by the recursion (PDR) converges to a solution of (P).

Proof. The sequence $\{(\mathbf{w}^k, \mathbf{b}^k)\}$ is the same as that generated by the Douglas-Rachford splitting scheme for $A = \partial f$ and $B = \partial[g \circ \mathbf{M}]$. Even though $(\mathbf{w}^0, \mathbf{b}^0)$ might not be in B, it is easily confirmed that $(\mathbf{w}^k, \mathbf{b}^k) \in B$ for all $k \geq 1$. As (P) is primal splittable and solvable, $\text{zer}(A+B) \neq \varnothing$. By Theorem 3.15, $\{\mathbf{t}^k\} = \{\mathbf{w}^k + \lambda\mathbf{b}^k\}$ converges to some element $\mathbf{t}^*$ of the nonempty set $T_\lambda^* = \{\ \mathbf{w} + \lambda\mathbf{b}\ \mid\ (\mathbf{w}, \mathbf{b}) \in B,\ (\mathbf{w}, -\mathbf{b}) \in A\}$. Let $(\mathbf{w}^*, \mathbf{b}^*)$ be the unique pair in B such that $\mathbf{w}^* + \lambda\mathbf{b}^* = \mathbf{t}^*$. applying the continuous operator $J_{\lambda B}$ to $\{\mathbf{t}^k\}$, we obtain $\mathbf{w}^k \to \mathbf{w}^*$. From the form of T_λ^*, $\mathbf{w}^*$ solves (P). ■

It is interesting to consider the special case where $\mathbf{M} = \mathbf{I}$ and $g = \delta_C$, $C \subseteq \mathbb{R}^n$ being some nonempty closed convex set. The condition $(\mathbf{w}^k, \mathbf{b}^k) \in B$ then reduces to $\mathbf{b}^k$ being normal to C at $\mathbf{w}^k$. Making the substitution $\mathbf{v}^k = \lambda\mathbf{b}^k$ while holding λ_k constant at $\lambda > 0$, one obtains the following, apparently original method:

$$\begin{aligned}
\mathbf{y}^{k+1} &= \arg\min_{\mathbf{y}} \left\{ f(\mathbf{y}) + \frac{1}{2\lambda} \| \mathbf{y} - (\mathbf{w}^k - \mathbf{v}^k) \|^2 \right\} \\
\mathbf{w}^{k+1} &= P_C(\mathbf{y}^{k+1} + \mathbf{v}^k) \\
\mathbf{v}^{k+1} &= \mathbf{v}^k + (\mathbf{y}^{k+1} - \mathbf{w}^{k+1}) \quad .
\end{aligned} \tag{X}$$

Proposition 3.41. Let $C \subseteq \mathbb{R}^n$ be a nonempty closed convex set, and let f be closed convex function attaining a minimum on C, such that $\text{ri}(\text{dom}\, f) \supseteq C$. Then for any $\lambda > 0$, and starting point $(\mathbf{w}^0, \mathbf{v}^0) \in \mathbb{R}^n \times \mathbb{R}^n$, the method (X) produces a sequence $\{\mathbf{w}^k\}$ converging to a point minimizing f over C.

Proof. The problem of minimizing f over C may be written

$$\text{minimize}_{\mathbf{x}\in\mathbb{R}^n}\ f(\mathbf{x}) + \delta_C(\mathbf{x}) \quad , \tag{PC}$$

which is the special case of (P) in which $g = \delta_C$ and $\mathbf{M} = \mathbf{I}$. The assumptions guarantee that this problem is solvable. Since C is nonempty, so is $\operatorname{ri}(\operatorname{dom} g) = \operatorname{ri} C \subseteq C$. Thus, $\operatorname{ri}(\operatorname{dom} f) \cap \operatorname{ri}(\operatorname{dom} g) \neq \varnothing$, and Proposition 3.23(i) implies that (PC) is primal splittable. Proposition 3.40 then implies the desired convergence of $\{\mathbf{w}^k\}$. ■

It is possible to derive variations of Proposition 3.41 depending on Proposition 3.23(ii-iv), rather than 3.23(i). The main point is that (X) is a variant of the gradient projection method that converges under less stringent conditions on f. Also note that in the case of $C = \mathbb{R}^n$, (X) reduces to Rockafellar's (1976b) proximal minimization algorithm. The method (X) could potentially be useful in situations where minimizing the sum of f and a quadratic term is not too difficult.

We now turn to dual algorithms for the general problem (P), that is, setting $\partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$. We restate the tight Douglas-Rachford iteration in a manner appropriate for a dual algorithm:

(a''') Given $(\mathbf{p}^k, \mathbf{z}^k) \in B$, find the unique $(\mathbf{q}^{k+1}, \mathbf{s}^{k+1}) \in A$ such that $\mathbf{q}^{k+1} + \lambda_k \mathbf{s}^{k+1} = \mathbf{p}^k - \lambda_k \mathbf{z}^k$

(b''') Find the unique $(\mathbf{p}^{k+1}, \mathbf{z}^{k+1}) \in B$ such that $\mathbf{p}^{k+1} + \lambda_k \mathbf{b}^{k+1} = \mathbf{q}^{k+1} + \lambda_k \mathbf{z}^k$.

Using Proposition 3.32(iv) with

$$\overline{\mathbf{p}} = \mathbf{p}^k - \lambda_k \mathbf{z}^k, \qquad \eta = \mathbf{z}^k, \qquad \phi = \mathbf{0}$$

shows that (a''') may be implemented, when $\mathbf{M}$ has full column rank, as

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle(\mathbf{p}^k - \lambda_k \mathbf{z}^k) + \lambda_k \mathbf{z}^k, \mathbf{M}\mathbf{x}\rangle + \frac{\lambda_k}{2}\|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2\}$$
$$\mathbf{q}^{k+1} = \mathbf{p}^k - \lambda_k \mathbf{z}^k + \lambda_k \mathbf{M}\mathbf{x}^{k+1}$$
$$\mathbf{s}^{k+1} = -\mathbf{M}\mathbf{x}^{k+1} \quad .$$

Proposition 3.32(iii) with

$$\bar{\mathbf{p}} = \mathbf{q}^{k+1} + \lambda_k \mathbf{z}^k = \mathbf{p}^k + \lambda_k \mathbf{M}\mathbf{x}^{k+1}, \qquad \zeta = \mathbf{M}\mathbf{x}^{k+1}, \qquad \theta = \mathbf{0}$$

then gives that (b''') can be written

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{p}^k + \lambda_k \mathbf{M}\mathbf{x}^{k+1} - \lambda_k \mathbf{M}\mathbf{x}^{k+1}, \mathbf{z}\rangle + \frac{\lambda_k}{2}\|\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}\|^2\}$$
$$\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda_k \mathbf{M}\mathbf{x}^{k+1} - \lambda \mathbf{z}^{k+1} \quad .$$

Combining these two calculations yields the *alternating direction method of multipliers* referred to in Chapter 2:

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x}\rangle + \frac{\lambda_k}{2}\|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2\}$$
$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z}\rangle + \frac{\lambda_k}{2}\|\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}\|^2\} \qquad \text{(ADMOM)}$$
$$\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda_k(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) \quad .$$

The connection between the alternating direction method of multipliers and the Douglas-Rachford splitting scheme was first discovered by Gabay (1983). Much as for the method (DPR), we offer the following interpretation in terms of the problem formulation (P'):

(I) Minimize the augmented Lagrangian for (P') with respect to **x** only.

(II) Minimize the augmented Lagrangian for (P') with respect to **z** only.

(III) Update the Lagrange multipliers.

Note that steps (I) and (II) do not generally suffice to minimize the augmented Lagrangian of (P'), yet one still updates the multipliers in step (III). In the conventional method of multipliers, one would replace steps (I) and (II) with a true minimization of the augmented Lagrangian.

We can now use Theorem 3.15 to give a convergence result for the alternating direction method of multipliers.

Proposition 3.42. Consider a convex program in the form (P),

$$\text{minimize}_{\mathbf{x}\in\mathbb{R}^n} \; f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \quad ,$$

possessing a Kuhn-Tucker pair. Suppose that the sequence $\{\lambda_k\}$ is constant at some $\lambda > 0$, and $\mathbf{M}$ has full column rank. Then for any starting point $(\mathbf{p}^0, \mathbf{z}^0) \in \mathbb{R}^m \times \mathbb{R}^m$, the alternating direction method of multipliers (ADMOM) produces sequences $\{\mathbf{x}^k\}$ and $\{\mathbf{p}^k\}$ that converge to solutions of (P) and its dual problem (D), respectively. Furthermore, $\mathbf{z}^k$ converges to $\mathbf{M}\mathbf{x}^*$, where $\mathbf{x}^*$ is the limit of $\{\mathbf{x}^k\}$.

Proof. The assumptions on (P) guarantee that $\text{zer}(A+B)$ is nonempty, as is $T_\lambda^* = \{\, \mathbf{p} + \lambda\mathbf{z} \mid (\mathbf{p}, \mathbf{z}) \in B, (\mathbf{p}, -\mathbf{z}) \in A \,\}$. Since (ADMOM) is an implementation of Douglas-Rachford splitting, we obtain that $\{\mathbf{t}^k\} = \{\mathbf{p}^k + \lambda\mathbf{z}^k\}$ converges to some element $\mathbf{t}^*$ of T_λ^*. Let $(\mathbf{p}^*, \mathbf{z}^*)$ be the unique pair in B such that $\mathbf{p}^* + \lambda\mathbf{z}^* = \mathbf{t}^*$. Applying the continuous operator $J_{\lambda B}$ to $\{\mathbf{t}^k\}$, we obtain $\mathbf{p}^k \to \mathbf{p}^*$ and $\mathbf{z}^k \to \mathbf{z}^*$. Using the convergence of $\{\mathbf{p}^k\}$ and $\{\mathbf{z}^k\}$, together with $\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda_k(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1})$ for all $k \geq 0$, we also obtain $\mathbf{M}\mathbf{x}^k \to \mathbf{z}^*$. Since the null space of $\mathbf{M}$ is $\{\mathbf{0}\}$, it follows that $\{\mathbf{x}^k\}$ converges to some $\mathbf{x}^*$, where $\mathbf{M}\mathbf{x}^* = \mathbf{z}^*$. From Proposition 3.32(iv), we note that

$$(\mathbf{x}^{k+1}, -\mathbf{M}^T\mathbf{q}^{k+1}) = (\mathbf{x}^{k+1}, -\mathbf{M}^T(\mathbf{p}^k - \lambda_k\mathbf{z}^k + \lambda_k\mathbf{M}\mathbf{x}^{k+1})) \in \partial f$$

for all $k \geq 0$. Taking limits and using Proposition 3.2, we have $(\mathbf{x}^*, -\mathbf{M}^T\mathbf{p}^*) \in \partial f$. Since

$$(\mathbf{p}^*, \mathbf{z}^*) = (\mathbf{p}^*, \mathbf{M}\mathbf{x}^*) \in B = \partial g^* = (\partial g)^{-1} ,$$

we also have $(\mathbf{M}\mathbf{x}^*, \mathbf{p}^*) \in \partial g$. So, $\mathbf{x}^*$ and $\mathbf{p}^*$ are jointly optimal by our Kuhn-Tucker theorem. ■

The Douglas-Rachford scheme displays a primal-dual symmetry very similar to that of the Peaceman-Rachford scheme.. Again, there are two ways of demonstrating the symmetry. The first is to complete the square in the two minimizations required by the alternating direction method of multipliers, thus removing the linear terms. One can then notice that the resulting method is identical to (PDR) except that λ_k is replaced throughout by $\frac{1}{\lambda_k}$. We give an alternate proof, very similar to our Peaceman-Rachford proof, that attempts to highlight some of the underlying structure.

Lemma 3.6. Let r be any positive real number and let A and B be maximal monotone operators on any Hilbert space $\mathcal{H}$. Let $A' = (-I)A^{-1}(-I)$ and $B' = B^{-1}$. Also let

$$\begin{aligned} G &= J_{rA}(2J_{rB} - I) + (I - J_{rB}) \\ G' &= J_{(1/r)A'}(2J_{(1/r)B'} - I) + (I - J_{(1/r)B'}) \quad . \end{aligned}$$

Then $G' = (\frac{1}{r}I)G(rI)$.

Proof. The identities of Proposition 3.34 state that

$$\begin{aligned} I - J_{(1/r)B'} &= (\tfrac{1}{r}I)J_{rB}(rI) \\ 2J_{(1/r)B'} - I &= (-\tfrac{1}{r}I)(2J_{rB} - I)(rI) \\ J_{(1/r)A'} &= (-\tfrac{1}{r}I)(I - J_{rA})(-rI) \quad . \end{aligned}$$

Then

$$
\begin{aligned}
G' &= (-\tfrac{1}{r}I)(I - J_{rA})(-rI)[(-\tfrac{1}{r}I)(2J_{rB} - I)(rI)] + (\tfrac{1}{r}I)J_{rB}(rI) \\
&= (-\tfrac{1}{r}I)(I - J_{rA})(2J_{rB} - I)(rI) + (\tfrac{1}{r}I)J_{rB}(rI) \\
&= (-\tfrac{1}{r}I)(2J_{rB} - I - J_{rA}(2J_{rB} - I))(rI) + (\tfrac{1}{r}I)J_{rB}(rI) \\
&= (\tfrac{1}{r}I)(J_{rA}(2J_{rB} - I) - 2J_{rB} + I)(rI) + (\tfrac{1}{r}I)J_{rB}(rI) \\
&= (\tfrac{1}{r}I)(J_{rA}(2J_{rB} - I) - 2J_{rB} + I + J_{rB})(rI) \\
&= (\tfrac{1}{r}I)(J_{rA}(2J_{rB} - I) - (J_{rB} - I))(rI) \\
&= (\tfrac{1}{r}I)G(rI) \ . \quad \blacksquare
\end{aligned}
$$

Proposition 3.43. The tight Douglas-Rachford splitting scheme displays the following primal-dual symmetry: let $\mathbf{M} = \mathbf{I}$ and suppose the primal Douglas-Rachford algorithm (PDR) generates the sequence $\{(\mathbf{w}^k, \mathbf{b}^k)\}$, starting from some $(\mathbf{w}^0, \mathbf{b}^0) \in \mathbb{R}^n \times \mathbb{R}^n$, while the alternating direction method of multipliers (ADMOM) generates the sequence $\{(\mathbf{p}^k, \mathbf{z}^k)\}$, starting from some $(\mathbf{p}^0, \mathbf{z}^0) \in \mathbb{R}^n \times \mathbb{R}^n$. Let $\{\lambda_k\}$ denote the stepsizes used in the primal, and $\{\lambda_k'\}$ the stepsizes used in the dual. Then if $\lambda_k = 1/\lambda_k'$ for all $k \geq 0$, $\mathbf{w}^0 = \mathbf{z}^0$, and $\mathbf{b}^0 = \mathbf{p}^0$, one has $\mathbf{w}^k = \mathbf{z}^k$ and $\mathbf{b}^k = \mathbf{p}^k$ for all $k \geq 0$.

Proof. Since $\mathbf{M} = \mathbf{I}$, the primal Douglas-Rachford algorithm uses the tight Douglas-Rachford splitting scheme on the operators $A = \partial f$ and $B = \partial g$, while the alternating direction method of multipliers applies it to

$$A' = \partial[f^* \circ (-\mathbf{I})] = (-\mathbf{I})A^{-1}(-\mathbf{I})$$

and $B' = \partial g^* = B^{-1}$. We now use induction to prove $\mathbf{w}^k = \mathbf{z}^k$ and $\mathbf{b}^k = \mathbf{p}^k$ for all $k \geq 0$. The basis of the induction, $\mathbf{w}^0 = \mathbf{z}^0$, and $\mathbf{b}^0 = \mathbf{p}^0$ is true by assumption. Fix any k and assume that $\mathbf{w}^k = \mathbf{z}^k$ and $\mathbf{b}^k = \mathbf{p}^k$. Set $r = \lambda_k$ and define G and G' as in Lemma 3.6:

$$
\begin{aligned}
G &= J_{rA}(2J_{rB} - I) + (I - J_{rB}) \\
G' &= J_{(1/r)A'}(2J_{(1/r)B'} - I) + (I - J_{(1/r)B'}) \quad .
\end{aligned}
$$

The procedure for obtaining $(\mathbf{w}^{k+1}, \mathbf{b}^{k+1})$ from $(\mathbf{w}^k, \mathbf{b}^k)$ in (PDR) may be expressed

$$\mathbf{t} = [J_{rA}(2J_{rB} - I) + (I - J_{rB})](\mathbf{w}^k + r\mathbf{b}^k) = G(\mathbf{w}^k + r\mathbf{b}^k)$$
$$\text{Find the unique } (\mathbf{w}^{k+1}, \mathbf{b}^{k+1}) \in B \text{ such that } \mathbf{w}^{k+1} + r\mathbf{b}^{k+1} = \mathbf{t} ,$$

while the procedure for obtaining $(\mathbf{p}^{k+1}, \mathbf{z}^{k+1})$ from $(\mathbf{p}^k, \mathbf{z}^k)$ in the alternating direction method of multipliers may be written as follows:

$$\mathbf{u} = [J_{(1/r)A'}(2J_{(1/r)B'} - I) + (I - J_{(1/r)B'})](\mathbf{p}^k + \tfrac{1}{r}\mathbf{z}^k) = G'(\mathbf{p}^k + \tfrac{1}{r}\mathbf{z}^k)$$
$$\text{Find the unique } (\mathbf{p}^{k+1}, \mathbf{z}^{k+1}) \in B' \text{ such that } \mathbf{p}^{k+1} + \tfrac{1}{r}\mathbf{z}^{k+1} = \mathbf{u} .$$

Using the identity $G' = (\frac{1}{r}I)G(rI)$ provided by the Lemma 3.6, the remainder of the induction step proceeds almost exactly as in Proposition 3.39. ■

Primal-dual symmetry reveals why it is generally more fruitful to examine dual applications of the Peaceman-Rachford and Douglas-Rachford splitting schemes than primal ones: when $\mathbf{M} \neq \mathbf{I}$, the primal schemes may present problems due to the presence of terms of the form $g(\mathbf{Mx})$ in some of the mimimands, but when $\mathbf{M} = \mathbf{I}$, they generate essentially the same algorithms as the dual approach. Nevertheless, there are some valid reasons to consider the primal schemes. First, they may lead to simpler derivations than the dual ones, with alternative convergence criteria. For example, the method (X) above uses $\mathbf{M} = \mathbf{I}$, and so could have been derived from the alternating direction method of multipliers. However, the primal derivation is more straightforward. It is also appropriate to examine the primal approach when one is using a splitting scheme that is not known to be primal-dual symmetric, as in the case of the forward-backward method. In such cases, the primal and dual approaches could yield truly distinct optimization algorithms.

Chapter 4

The Splitting Operator

The previous chapter presented a unified view of monotone operators, splitting methods, and decomposition methods for convex programming. From now on, we will concentrate exclusively on the Douglas-Rachford splitting scheme. This chapter will introduce the *splitting operator*, a concept that further unifies the area of Douglas-Rachford operator splitting and convex programming decomposition. The splitting operator brings two main benefits: first, it demonstrates that the Douglas-Rachford splitting scheme is actually an application of the proximal point algorithm, allowing much proximal point theory to be carried over into the Douglas-Rachford realm; second, it allows us to demonstrate that Spingarn's (1983, 1985b) *method of partial inverses* is a special case of Douglas-Rachford splitting. Spingarn (1985) has derived a parallel algorithm for block-separable convex programming from the partial inverse method, which serves as some indication of the applicability of operator splitting to parallel optimization. This chapter will also demonstrate that Gol'shtein's (1987) "general decompositional method" for sums of monotone operators, from which Gol'shtein (1985, 1986) has derived parallel optimization algorithms, is again an application of Douglas-Rachford splitting. This fact can be shown without the aid of the splitting operator; we mention it in this chapter primarily because of its close relationship with Spingarn's work.

Preliminary versions of some of the results of this chapter have already appeared in Eckstein (1988).

4.1. Defining the Splitting Operator

Given the background material presented in Chapter 3, the idea of the splitting operator should now develop fairly naturally. Recall that in Section 3.4.4, we based the proof of convergence for the Douglas-Rachford splitting scheme on the (unproved) assertion that for any maximal monotone operators A and B, the operator

$$G_{\lambda,A,B} = J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B}) \ .$$

is firmly nonexpansive. Note also that as $J_{\lambda A}$ and $J_{\lambda B}$ have full domain, so does $G_{\lambda,A,B}$. Corollary 3.6.1 then indicates that since $G_{\lambda,A,B}$ is supposed to be firmly nonexpansive with full domain, then the operator

$$S_{\lambda,A,B} \ = \ (G_{\lambda,A,B})^{-1} - I$$

ought to be maximal monotone. We begin by studying the properties of this operator. We first seek a set-theoretical expression for $S_{\lambda,A,B}$. Let λ be any positive scalar, let $\mathcal{H}$ be any Hilbert space over the reals, and suppose that A and B are maximal monotone operators on $\mathcal{H}$. Consider any $z \in \mathcal{H}$, and recall that there is some unique $(u, b) \in B$ such that $u + \lambda b = z$. Then

$$G_{\lambda,A,B}(z) \ = \ [J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})](u + \lambda b) \ = \ J_{\lambda A}(u - \lambda b) + \lambda b \ .$$

We thus arrive at the following expression for $G_{\lambda,A,B}$:

$$G_{\lambda,A,B} \ = \ \{(u + \lambda b, v + \lambda b) \mid (u, b) \in B, (v, a) \in A, v + \lambda a = u - \lambda b\} \ .$$

A simple manipulation of this representation provides an expression for $(G_{\lambda,A,B})^{-1} - I$:

$$(G_{\lambda,A,B})^{-1} \ = \ \{(v + \lambda b, u + \lambda b) \mid (u, b) \in B, (v, a) \in A, v + \lambda a = u - \lambda b\}$$

$$(G_{\lambda,A,B})^{-1} - I \ = \ \{(v + \lambda b, u - v) \mid (u, b) \in B, (v, a) \in A, v + \lambda a = u - \lambda b\} \ .$$

Definition 4.1. Given any Hilbert space $\mathcal{H}$, $\lambda > 0$, and maximal monotone operators A,B: $\mathcal{H} \rightrightarrows \mathcal{H}$, the *splitting operator* of A and B with respect to λ is

$$S_{\lambda,A,B} = \{(v+\lambda b, u-v) \mid (u, b) \in B, (v, a) \in A, v+\lambda a = u-\lambda b\} .$$

Given the discussion above, the following result needs no further proof:

Proposition 4.1. Given $\lambda > 0$ and operators A and B on a linear space,

$$(I + S_{\lambda,A,B})^{-1} = G_{\lambda,A,B} = \{(u+\lambda b, v+\lambda b) \mid (u, b) \in B, (v, a) \in A, v+\lambda a = u-\lambda b\}$$

$$(G_{\lambda,A,B})^{-1} - I = S_{\lambda,A,B} .$$

We now directly establish the maximal monotonicity of $S_{\lambda,A,B}$.

Proposition 4.2. If A and B are monotone then $S_{\lambda,A,B}$ is monotone. If A and B are maximal monotone, then $S_{\lambda,A,B}$ is maximal monotone.

Proof. First we show that $S_{\lambda,A,B}$ is monotone. Let $u, b, v, a, u', b', v', a' \in \mathcal{H}$ be such that $(u, b), (u', b') \in B$, $(v, a), (v', a') \in A$, $v+\lambda a = u-\lambda b$, and $v'+\lambda a' = u'-\lambda b'$. Then we have that

$$a = \tfrac{1}{\lambda}(u-v) - b \qquad a' = \tfrac{1}{\lambda}(u'-v') - b' ,$$

and

$$\begin{aligned}
&\langle (v'+\lambda b') - (v+\lambda b), (u'-v') - (u-v)\rangle \\
&\quad = \lambda\langle (v'+\lambda b') - (v+\lambda b), \lambda^{-1}(u'-v') - b' - \lambda^{-1}(u-v) + b\rangle \\
&\qquad + \lambda\langle (v'+\lambda b') - (v+\lambda b), b'-b\rangle \\
&\quad = \lambda\langle v'-v, \lambda^{-1}(u'-v') - b' - \lambda^{-1}(u-v) + b\rangle \\
&\qquad + \lambda^2\langle b'-b, \lambda^{-1}(u'-v') - b' - \lambda^{-1}(u-v) + b\rangle \\
&\qquad + \lambda\langle v'-v, b'-b\rangle + \lambda^2\langle b'-b, b'-b\rangle
\end{aligned}$$

$$
\begin{aligned}
&= \lambda\langle v'-v, a'-a\rangle + \lambda\langle b'-b, u'-u\rangle - \lambda\langle b'-b, v'-v\rangle - \lambda^2\langle b'-b, b'-b\rangle \\
&\quad + \lambda\langle v'-v, b'-b\rangle + \lambda^2\langle b'-b, b'-b\rangle \\
&= \lambda\langle v'-v, a'-a\rangle + \lambda\langle b'-b, u'-u\rangle .
\end{aligned}
$$

By the monotonicity of A and B, the two terms in the final line are nonnegative, so we obtain that $\langle (v'+\lambda b') - (v+\lambda b), (u'-v') - (u-v)\rangle \geq 0$, and $S_{\lambda,A,B}$ is monotone. It remains to show that $S_{\lambda,A,B}$ is maximal in the case that A and B are. By Theorem 3.6, we only need to show that $(I + S_{\lambda,A,B})^{-1} = G_{\lambda,A,B} = J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})$ has full domain. This is indeed the case, as $J_{\lambda A}$ and $J_{\lambda B}$ are defined everywhere. ■

Since $G_{\lambda,A,B} = (I + S_{\lambda,A,B})^{-1}$, Theorem 3.6 also gives us:

Corollary 4.2.1. $G_{\lambda,A,B}$ is firmly nonexpansive (and has full domain).

This corollary completes the heretofore deferred proof of Lemma 3.2.

Examining the splitting operator itself provides a less circuitous explanation of why $S_{\lambda,A,B}$ might be useful in finding a zero of $A+B$. We may rephrase the problem of locating a zero of $A+B$ as that of finding $(u, b) \in B$, $(v, a) \in A$ such that

$$
\begin{aligned}
v &= u \\
a &= -b \ .
\end{aligned}
$$

For any real number $\lambda > 0$, an equivalent system of equations is

$$
\begin{aligned}
u - v &= 0 \\
v + \lambda a &= u - \lambda b \ .
\end{aligned}
$$

One way to determine such u, b, v and a is to find a zero of the operator

$$\{(v+\lambda b,\, u-v) \mid (u, b) \in B,\, (v, a) \in A,\, v+\lambda a = u-\lambda b\}\ ,$$

which is precisely $S_{\lambda,A,B}$. The choice of $v+\lambda b$ for the first coordinate may seem arbitrary, but as we have seen above, this choice makes the operator monotone.

We formalize the above discussion as follows:

Proposition 4.3. Given any Hilbert space $\mathcal{H}$, $\lambda > 0$, and maximal monotone operators $A,B: \mathcal{H} \rightrightarrows \mathcal{H}$,

$$\mathrm{zer}(S_{\lambda,A,B}) = T_\lambda^* = \{\, u+\lambda b \mid b \in Bu,\ -b \in Au \,\} \subseteq \{\, u+\lambda b \mid u \in \mathrm{zer}(A+B),\, b \in Bu\}\ .$$

Proof. Let $S = S_{\lambda,A,B}$. We wish to show that $\mathrm{zer}(S)$ is equal to T_λ^*. Let $z \in \mathrm{zer}(S)$. Then there exist some $u, b, v, a \in \mathcal{H}$ such that $v+\lambda b = z$, $u - v = 0$, $(u, b) \in B$, and $(v, a) \in A$. So,

$$u - v = 0 \ \Rightarrow\ u = v \ \Rightarrow\ \lambda a = -\lambda b \ \Rightarrow\ a = -b\ ,$$

and we have $u+\lambda b = z$, $(u, b) \in B$, and $(u, -b) \in A$, hence $z \in T_\lambda^*$. Conversely, if $z \in T_\lambda^*$, then $z = u+\lambda b$, $b \in Bu$, and $-b \in Au$. Setting $u = v$ and $a = -b$, we see that $(z, 0) \in S$. Finally, the inclusion $T_\lambda^* \subseteq \{\, u+\lambda b \mid u \in \mathrm{zer}(A+B),\, b \in Bu\}$ follows because $b \in Bu$ and $-b \in Au$ imply that $u \in \mathrm{zer}(A+B)$. ■

This result could also have been derived from Propositions 3.9 and 3.20. We now return to considering the ramifications of $(I + S_{\lambda,A,B})^{-1} = G_{\lambda,A,B}$.

Proposition 4.4. With fixed stepsize $\lambda > 0$, the Douglas-Rachford iteration $t^{k+1} = [J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})]t^k$ is equivalent to applying the proximal point algorithm to the maximal monotone operator $S_{\lambda,A,B}$, with the proximal point stepsize γ_k fixed at 1.

Proof. The Douglas-Rachford iteration is $t^{k+1} = G_{\lambda,A,B}(t^k)$, which is just $t^{k+1} = (I + S_{\lambda,A,B})^{-1}(t^k)$. ■

This result gives some intuition as to why the convergence properties of the Douglas-Rachford method are so much more general than those of other splitting schemes — the Douglas-Rachford iteration is actually an application of the proximal point algorithm, which has very general convergence properties.

At this point, it is tempting to consider applying the proximal point algorithm to the splitting operator with stepsize sequences other than $\gamma_k = 1$ for all k. Consider, for any $\gamma > 0$, trying to execute the step $t^{k+1} = (I + \gamma S_{\lambda,A,B})^{-1}(t^k)$. A straightforward calculation yields that

$$(I + \gamma S_{\lambda,A,B})^{-1} = \{ ((1-\gamma)v + \gamma u + \lambda b,\ v + \lambda b) \mid (u, b) \in B,\ (v, a) \in A,\ v + \lambda a = u - \lambda b \}.$$

Thus, to calculate $(I + \gamma S_{\lambda,A,B})^{-1}(z)$, one must find $(u, b) \in B$ and $(v, a) \in A$ such that

$$(1-\gamma)v + \gamma u + \lambda b = z \qquad a = \frac{1}{\lambda}(u - v) - b) \ .$$

Alternatively, we may state the problem as that of finding $u, v \in \mathcal{H}$ such that

$$z - (\gamma u + (1-\gamma)v) \in \lambda B u \qquad -z + ((1+\gamma)u - \gamma v) \in \lambda A v \ .$$

This does not appear to be a particularly easy problem. Specifically, it does not appear to be any less difficult than the calculation of $J_{\lambda(A+B)}$ at an arbitrary point z, which, when using a splitting algorithm, we are expressly trying to avoid. By comparison, that calculation involves finding $(u, b) \in B$ such that $(u, \lambda^{-1}(z-u) - b) \in A$.

Consider, however, what happens when one fixes γ at 1. Then one has only to find

$$(u, b) \in B \quad \text{such that} \quad u+\lambda b = z$$
$$(v, a) \in A \quad \text{such that} \quad v+\lambda a = 2u - z = u - \lambda b \ .$$

The conditions $(u, b) \in B$, $u+\lambda b = z$ uniquely determine $u = J_{\lambda,B}(z)$ and $b = \frac{1}{\lambda}(z - u)$ independently of v. Once u is known, then v is likewise uniquely determined by $u = J_{\lambda,A}(u - \lambda b)$. We have thus achieved a *decomposition* in which the calculation of $J_{S_{\lambda,A,B}} = (I + S_{\lambda,A,B})^{-1}$ is replaced by separate, sequential evaluations of $J_{\lambda A} = (I + \lambda A)^{-1}$ and $J_{\lambda B} = (I + \lambda B)^{-1}$. This procedure is essentially the tight Douglas-Rachford calculation (a')-(b') of Section 3.4.4. It seems that keeping $\gamma = 1$ at all times is critical to the decomposition. Spingarn (1985) has already recognized this phenomenon, but in the more restrictive context of his method of partial inverses. The next section will show that Spingarn's method is a special case of Douglas-Rachford splitting.

A more fruitful generalization of Douglas-Rachford splitting may be obtained, however, from Theorem 3.9. This theorem allows us to introduce two additional elements into Douglas-Rachford splitting: *under- or over-relaxation* and *approximate computation*. We now restrict ourselves to finite dimension.

Proposition 4.5. Given some $\mathbf{t}^0 \in \mathbb{R}^n$, $\lambda > 0$, and maximal monotone operators $A,B: \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ such that $\text{zer}(A+B) \neq \emptyset$, let $\{\mathbf{t}^k\}_{k=0}^{\infty} \subseteq \mathbb{R}^n$, $\{\mathbf{u}^k\}_{k=0}^{\infty} \subseteq \mathbb{R}^n$, $\{\mathbf{v}^k\}_{k=1}^{\infty} \subseteq \mathbb{R}^n$, $\{\alpha_k\}_{k=0}^{\infty} \subseteq [0, \infty)$, $\{\beta_k\}_{k=0}^{\infty} \subseteq [0, \infty)$, and $\{\rho_k\}_{k=0}^{\infty} \subseteq (0, 2)$ conform to the following conditions:

$$\| \mathbf{u}^k - J_{\lambda B}(\mathbf{t}^k) \| \leq \beta_k \qquad \forall\, k \geq 0 \qquad \text{(T1)}$$

$$\| \mathbf{v}^{k+1} - J_{\lambda A}(2\mathbf{u}^k - \mathbf{t}^k) \| \leq \alpha_k \qquad \forall\, k \geq 0 \qquad \text{(T2)}$$

$$\mathbf{t}^{k+1} = \mathbf{t}^k + \rho_k(\mathbf{v}^{k+1} - \mathbf{u}^k) \qquad \forall\, k \geq 0 \qquad \text{(T3)}$$

$$\sum_{k=0}^{\infty} \alpha_k \;<\; \infty$$

$$\sum_{k=0}^{\infty} \beta_k \;<\; \infty$$

$$0 \;<\; \inf_{k \geq 0} \rho_k \;\leq\; \sup_{k \geq 0} \rho_k \;<\; 2 \quad .$$

Then $\{\mathbf{t}^k\}$converges to some element of $T_\lambda^* = \{\, \mathbf{u}+\lambda\mathbf{b} \mid \mathbf{b} \in B\mathbf{u},\ -\mathbf{b} \in A\mathbf{u} \,\}$.

Proof. Fix any k. Then $\|\, \mathbf{u}^k - J_{\lambda B}(\mathbf{t}^k) \,\| \;\leq\; \beta_k$ implies that

$$\|\, (2\mathbf{u}^k - \mathbf{t}^k) - (2J_{\lambda B} - I)(\mathbf{t}^k) \,\| \;\leq\; 2\beta_k \quad .$$

Since $J_{\lambda A}$ is nonexpansive,

$$\|\, J_{\lambda A}(2\mathbf{u}^k - \mathbf{t}^k) - J_{\lambda A}(2J_{\lambda B} - I)(\mathbf{t}^k) \,\| \;\leq\; 2\beta_k \quad ,$$

and so

$$\|\, \mathbf{v}^{k+1} - J_{\lambda A}(2J_{\lambda B} - I)(\mathbf{t}^k) \,\| \;\leq\; 2\beta_k + \alpha_k$$

$$\|\, (\mathbf{v}^{k+1} + \mathbf{t}^k - \mathbf{u}^k) - [J_{\lambda A}(2J_{\lambda B} - I) + (I - J_{\lambda B})](\mathbf{t}^k) \,\| \;\leq\; 3\beta_k + \alpha_k \quad .$$

Let $\varepsilon_k = 3\beta_k + \alpha_k$ for all k. Then

$$\sum_{k=0}^{\infty} \varepsilon_k \;=\; 3\sum_{k=0}^{\infty} \beta_k + \sum_{k=0}^{\infty} \alpha_k \;<\; \infty \quad .$$

We also have

$$\mathbf{t}^{k+1} \;=\; \mathbf{t}^k + \rho_k(\mathbf{v}^{k+1} - \mathbf{u}^k) \;=\; (1 - \rho_k)\mathbf{t}^k + \rho_k(\mathbf{v}^{k+1} + \mathbf{t}^k - \mathbf{u}^k) \quad .$$

Thus, letting $\mathbf{y}^k = \mathbf{v}^{k+1} + \mathbf{t}^k - \mathbf{u}^k$, we have

$$0 < \inf_{k \geq 0} \rho_k \leq \sup_{k \geq 0} \rho_k < 2$$

$$\sum_{k=0}^{\infty} \varepsilon_k < +\infty \quad ,$$

$$\|\mathbf{y}^k - G_{\lambda,A,B}\mathbf{t}^k\| \leq \varepsilon_k \qquad \forall\, k \geq 0$$

$$\mathbf{t}^{k+1} = (1-\rho_k)\mathbf{t}^k + \rho_k \mathbf{y}^k \qquad \forall\, k \geq 0 \quad .$$

Theorem 3.9 then guarantees that $\{\mathbf{t}^k\}$ converges to a zero of $S_{\lambda,A,B}$, that is, to a member of $T\lambda^*$. ■

It will be useful to have an expression for the method of Proposition 4.5 that is similar to the description of the tight Douglas-Rachford scheme (a')-(b') of Section 3.4.4.

Proposition 4.6. Let $A,B: \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ be maximal monotone operators, and let $\{\alpha_k\}_{k=0}^{\infty} \subseteq [0, \infty)$, $\{\beta_k\}_{k=1}^{\infty} \subseteq [0, \infty)$, and $\{\rho_k\}_{k=0}^{\infty} \subseteq (0, 2)$ conform to

$$\sum_{k=0}^{\infty} \alpha_k < \infty \qquad \sum_{k=1}^{\infty} \beta_k < \infty \qquad 0 < \inf_{k \geq 0} \rho_k \leq \sup_{k \geq 0} \rho_k < 2 \quad .$$

Let $\{\mathbf{u}^k\}_{k=0}^{\infty}$, $\{\mathbf{b}^k\}_{k=0}^{\infty}$, $\{\mathbf{v}^k\}_{k=1}^{\infty}$, $\{\mathbf{a}^k\}_{k=1}^{\infty} \subseteq \mathbb{R}^n$ be sequences meeting the following conditions, for some fixed $\lambda > 0$ and all $k \geq 0$:

$$\mathbf{v}^{k+1} + \lambda \mathbf{a}^{k+1} = \mathbf{u}^k - \lambda \mathbf{b}^k$$

$$\|\mathbf{v}^{k+1} - J_{\lambda A}(\mathbf{u}^k - \lambda \mathbf{b}^k)\| \leq \alpha_k$$

$$\mathbf{u}^{k+1} + \lambda \mathbf{b}^{k+1} = \rho_k \mathbf{v}^{k+1} + (1-\rho_k)\mathbf{u}^k + \lambda \mathbf{b}^k$$

$$\|\mathbf{u}^{k+1} - J_{\lambda B}(\rho_k \mathbf{v}^{k+1} + (1-\rho_k)\mathbf{u}^k + \lambda \mathbf{b}^k)\| \leq \beta_{k+1} \quad .$$

Then $\{(\mathbf{u}^k, \mathbf{b}^k)\}$ converges to some $(\mathbf{u}^*, \mathbf{b}^*) \in B$ such that $(\mathbf{u}^*, -\mathbf{b}^*) \in A$.

Proof. Let β_0 be some very large number, and let $\mathbf{t}^k = \mathbf{u}^k + \lambda \mathbf{b}^k$ for all $k \geq 0$. Our aim is to

show that $\{\mathbf{t}^k\}$, $\{\mathbf{u}^k\}$ and $\{\mathbf{v}^k\}$ evolve according to the hypotheses of Proposition 4.5. Now, for any k,

$$\|\mathbf{v}^{k+1} - J_{\lambda A}(\mathbf{u}^k - \lambda\mathbf{b}^k)\| = \|\mathbf{v}^k - J_{\lambda A}(2\mathbf{u}^k - \mathbf{t}^k)\| \leq \alpha_k \quad ,$$

giving hypothesis (T1) of Proposition 4.5. We also have

$$\mathbf{t}^{k+1} = \mathbf{u}^{k+1} + \lambda\mathbf{b}^{k+1} = \rho_k\mathbf{v}^{k+1} + (1-\rho_k)\mathbf{u}^k + \lambda\mathbf{b}^k = \mathbf{t}^k + \rho_k(\mathbf{v}^{k+1} - \mathbf{u}^k) \quad ,$$

giving (T2). Finally,

$$\|\mathbf{u}^{k+1} - J_{\lambda B}(\rho_k\mathbf{v}^{k+1} + (1-\rho_k)\mathbf{u}^k + \lambda\mathbf{b}^k)\| = \|\mathbf{u}^{k+1} - J_{\lambda B}(\mathbf{t}^{k+1})\| \leq \beta_{k+1} \quad ,$$

gives (T3) except for the case of $k = 0$. That case can be dispensed with by assuming β_0 is a sufficiently large number. Proposition 4.5 then gives that $\mathbf{t}^k \to \mathbf{t}^* \in T_\lambda^*$. Applying $J_{\lambda B}$ to this sequence and using $\|\mathbf{u}^{k+1} - J_{\lambda B}(\mathbf{t}^{k+1})\| \leq \beta_{k+1} \to 0$ gives that $\mathbf{u}^k \to \mathbf{u}^* \in \text{zer}(A+B)$, and $\mathbf{b}^k \to \mathbf{b}^* \in B\mathbf{u}^* \cap (-A\mathbf{u}^*)$. ■

We can immediately apply the generalized version of Douglas-Rachford splitting to convex programming.

Proposition 4.7 (The generalized alternating direction method of multipliers). Consider a convex program in the form (P), minimize $_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) + g(\mathbf{M}\mathbf{x})$, where $\mathbf{M}$ has full column rank. Assume that (P) possesses a Kuhn-Tucker pair. Let $\mathbf{p}^0, \mathbf{z}^0 \in \mathbb{R}^m$, and suppose we are given $\lambda > 0$ and

$$\{\mu_k\}_{k=0}^{\infty} \subseteq [0, \infty) \,, \qquad \sum_{k=0}^{\infty} \mu_k < \infty$$

$$\{\omega_k\}_{k=0}^{\infty} \subseteq [0, \infty) \,, \qquad \sum_{k=0}^{\infty} \omega_k < \infty$$

$$\{\rho_k\}_{k=0}^{\infty} \subseteq (0, 2), \qquad 0 < \inf_{k \geq 0} \rho_k \leq \sup_{k \geq 0} \rho_k < 2 \quad .$$

Suppose $\{\mathbf{x}^k\}_{k=1}^{\infty}$, $\{\mathbf{z}^k\}_{k=0}^{\infty}$, and $\{\mathbf{p}^k\}_{k=0}^{\infty}$ conform, for all k, to

$$\begin{aligned}
&\| \mathbf{x}^{k+1} - \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle + \tfrac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z}^k \|^2\} \| \leq \mu_k \\
&\| \mathbf{z}^{k+1} - \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z} \rangle + \tfrac{\lambda}{2} \| \rho_k \mathbf{M}\mathbf{x}^{k+1} + (1 - \rho_k)\mathbf{z}^k - \mathbf{z} \|^2\} \| \leq \omega_k \\
&\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda(\rho_k \mathbf{M}\mathbf{x}^{k+1} + (1 - \rho_k)\mathbf{z}^k - \mathbf{z}^{k+1}) \quad .
\end{aligned}$$

Then $\{\mathbf{x}^k\}$ converges to a solution of (P), and $\{\mathbf{p}^k\}$ converges to a solution of the dual problem (D). Furthermore, $\{\mathbf{z}^k\}$ converges to $\mathbf{M}\mathbf{x}^*$, where $\mathbf{x}^*$ is the limit of $\{\mathbf{x}^k\}$.

Proof. Let

$$\begin{aligned}
\mathbf{t}^k &= \mathbf{p}^k + \lambda \mathbf{z}^k && \forall\, k \geq 0 \\
\mathbf{q}^k &= \mathbf{p}^k + \lambda(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^k) && \forall\, k \geq 0 \\
A &= \partial[f^* \circ (-\mathbf{M}^T)] \\
B &= \partial g^* \\
\alpha_k &= \lambda \|\mathbf{M}\| \mu_k && \forall\, k \geq 0 \\
\beta_0 &= \| \mathbf{p}^0 - J_{\lambda B}(\mathbf{p}^0 + \lambda \mathbf{z}^0) \| \\
\beta_k &= \lambda \omega_k && \forall\, k \geq 1 \quad ,
\end{aligned}$$

where $\|\mathbf{M}\|$ denotes the l_2-norm of the matrix $\mathbf{M}$,

$$\|\mathbf{M}\| = \sup_{\mathbf{x} \neq \mathbf{0}} \left\{ \frac{\|\mathbf{M}\mathbf{x}\|}{\|\mathbf{x}\|} \right\} \quad .$$

We wish to establish that the following hold for all $k \geq 0$:

$$\| \mathbf{p}^k - J_{\lambda B}(\mathbf{t}^k) \| \leq \beta_k \tag{Y1}$$

$$\| \mathbf{q}^k - J_{\lambda A}(2\mathbf{p}^k - \mathbf{t}^k) \| \leq \alpha_k \tag{Y2}$$

$$\mathbf{t}^{k+1} = \mathbf{t}^k + \rho_k(\mathbf{q}^k - \mathbf{p}^k) \quad . \tag{Y3}$$

For $k = 0$, (Y1) is valid by the choice of β_0. Now suppose (Y1) holds for some k; we show that (Y2) also holds for k. By Proposition 3.32(iv) with $\eta = \lambda \mathbf{z}^k$, we have that $\tilde{\mathbf{p}}^k = J_{\lambda A}(2\mathbf{p}^k - \mathbf{t}^k) = J_{\lambda A}(\mathbf{p}^k - \lambda \mathbf{z}^k)$ may be computed via

$$\begin{aligned} \bar{\mathbf{x}}^k &= \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle + \tfrac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z}^k \|^2 \} \\ \tilde{\mathbf{p}}^k &= (\mathbf{p}^k - \lambda \mathbf{z}^k) + \lambda \mathbf{M} \bar{\mathbf{x}}^k \quad . \end{aligned}$$

Thus, from

$$\begin{aligned} \| \mathbf{x}^{k+1} - \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle + \tfrac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z}^k \|^2 \} \| &\leq \mu_k \\ \mathbf{q}^k &= \mathbf{p}^k + \lambda(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^k) \quad , \end{aligned}$$

we obtain

$$\| \mathbf{x}^k - \bar{\mathbf{x}}^k \| \leq \mu_k$$

$$\| \mathbf{q}^k - \tilde{\mathbf{p}}^k \| \leq \lambda \| \mathbf{M} \| \mu_k \quad ,$$

establishing (Y2) for k.

Suppose that (Y1) and (Y2) hold for some k. We now show that (Y3) holds for k and (Y1) holds for k+1. Let

$$\begin{aligned} \mathbf{s}^k &= \mathbf{t}^k + \rho_k(\mathbf{q}^k - \mathbf{p}^k) \\ &= \mathbf{p}^k + \mathbf{z}^k + \lambda \rho_k(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^k) \\ &= \mathbf{p}^k + \lambda(\rho_k \mathbf{M}\mathbf{x}^{k+1} + (1 - \rho_k)\mathbf{z}^k) \quad . \end{aligned}$$

To calculate $\tilde{\mathbf{s}}^k = J_{\lambda B}(\mathbf{s}^k)$, one could perform, using Proposition 3.32(iii) with $\zeta = \rho_k \mathbf{M}\mathbf{x}^{k+1} + (1 - \rho_k)\mathbf{z}^k$,

$$\bar{\mathbf{z}}^k = \arg\min_{\mathbf{z}} \{ g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z} \rangle + \frac{\lambda}{2} \| (\rho_k \mathbf{M}\mathbf{x}^{k+1} + (1-\rho_k)\mathbf{z}^k) - \mathbf{z} \|^2 \}$$
$$\tilde{\mathbf{s}}^k = \mathbf{p}^k + \lambda(\rho_k \mathbf{M}\mathbf{x}^{k+1} + (1-\rho_k)\mathbf{z}^k - \bar{\mathbf{z}}^k) \quad .$$

The condition on $\mathbf{z}^{k+1}$ is just $\| \mathbf{z}^{k+1} - \bar{\mathbf{z}}^k \| \leq \omega_k$, so $\| \mathbf{p}^{k+1} - \tilde{\mathbf{s}}^k \| \leq \lambda\omega_k$. We also have

$$\begin{aligned} \mathbf{t}^{k+1} &= \mathbf{p}^{k+1} + \lambda \mathbf{z}^{k+1} \\ &= \mathbf{p}^k + \lambda(\rho_k \mathbf{M}\mathbf{x}^{k+1} + (1-\rho_k)\mathbf{z}^k - \mathbf{z}^{k+1}) + \lambda \mathbf{z}^{k+1} \\ &= \mathbf{p}^k + \lambda(\rho_k \mathbf{M}\mathbf{x}^{k+1} + (1-\rho_k)\mathbf{z}^k) \\ &= \mathbf{s}^k \quad . \end{aligned}$$

Thus, (Y3) holds for k, and (Y1) holds for $k+1$ by $\| \mathbf{p}^{k+1} - \tilde{\mathbf{s}}^k \| \leq \lambda\omega_k$. By induction, then, (Y1)-(Y3) hold for all k. The summability of $\{\mu_k\}$ and $\{\omega_k\}$ implies the summability of $\{\beta_k\}$ and $\{\alpha_k\}$, so by Proposition 4.5, $\{\mathbf{t}^k\}$ converges to some element $\mathbf{t}^*$ of $T_\lambda^* = \{ \mathbf{p} + \lambda\mathbf{z} \mid \mathbf{z} \in B\mathbf{p}, \ -\mathbf{z} \in A\mathbf{p} \}$. Applying the continuous operator $J_{\lambda B}$ to $\{\mathbf{t}^k\}$ and using (Y1), we obtain $\mathbf{p}^k \to \mathbf{p}^*$ and $\mathbf{z}^k \to \mathbf{z}^*$, where $(\mathbf{p}^*, \mathbf{z}^*) \in B$ and $\mathbf{p}^* + \lambda\mathbf{z}^* = \mathbf{t}^*$. By rearranging the multiplier update formula, we have

$$(\mathbf{p}^{k+1} - \mathbf{p}^k) + (\mathbf{z}^{k+1} - \mathbf{z}^k) = \lambda\rho_k(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^k)$$

for all $k \geq 0$. Taking limits and using that ρ_k is bounded away from zero, we obtain that $(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^k) \to \mathbf{0}$, hence $\mathbf{M}\mathbf{x}^k \to \mathbf{z}^*$. As $\mathbf{M}$ has full column rank, $\mathbf{x}^k \to \mathbf{x}^*$, where $\mathbf{M}\mathbf{x}^* = \mathbf{z}^*$. We thus have $(\mathbf{p}^*, \mathbf{z}^*) = (\mathbf{p}^*, \mathbf{M}\mathbf{x}^*) \in B = \partial g^*$, and so $(\mathbf{M}\mathbf{x}^*, \mathbf{p}^*) \in \partial g$. Now, from Proposition 3.32(iv) we have that $(-\mathbf{M}^T\tilde{\mathbf{p}}^k, \bar{\mathbf{x}}^k) \in \partial f$ for all k. Using that

$$0 \leq \| \mathbf{q}^k - \tilde{\mathbf{p}}^k \| = \| \mathbf{p}^k + \lambda(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^k) - \tilde{\mathbf{p}}^k \| \leq \lambda \|\mathbf{M}\| \mu_k \to 0 \quad ,$$

we have by taking limits that $\tilde{\mathbf{p}}^k \to \mathbf{p}^*$, and using that $\| \mathbf{x}^k - \bar{\mathbf{x}}^k \| \leq \mu_k \to 0$, we also have $\bar{\mathbf{x}}^k \to \mathbf{x}^*$. Therefore $(-\mathbf{M}^T\mathbf{p}^*, \mathbf{x}^*) \in \partial f$ by Proposition 3.2, and $(\mathbf{x}^*, \mathbf{p}^*)$ is a Kuhn-Tucker pair for (P). ■

It should be noted that Glowinski and Le Tallec (1987) have also introduced over- and under-relaxation factors into the alternating direction method of multipliers, but in a different way.

Another benefit of knowing the relationship between the proximal point algorithm and Douglas-Rachford splitting is that it becomes possible to say what happens in Douglas-Rachford methods when $A+B$ has no zeroes.

Proposition 4.8. Keeping the stepsizes constant at $\lambda > 0$, suppose one applies the Douglas-Rachford splitting method $\mathbf{t}^{k+1} = G_{\lambda,A,B}(\mathbf{t}^k)$ in a situation where $\mathrm{zer}(A+B)$ is empty. Then $\{\mathbf{t}^k\}$ is an unbounded sequence.

Proof. If $\mathrm{zer}(A+B) = \varnothing$, then $\mathrm{zer}(S_{\lambda,A,B}) = T_\lambda{}^*$ is also empty. As the the Douglas-Rachford method is just the proximal point algorithm applied to $S_{\lambda,A,B}$, Theorem 3.8 implies that $\|\mathbf{t}^k\| \to \infty$. ■

Proposition 4.9. Suppose one applies the alternating direction method of multipliers, using a constant stepsize $\lambda > 0$, to a convex program of the form (P),

$$\text{minimize}_{\ \mathbf{x}\in\mathbb{R}^n}\ f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \quad ,$$

which is not solvable, even though $\mathbf{M}$ has full column rank. Then either $\|\mathbf{p}^k\| \to \infty$, or $\|\mathbf{z}^k\| \to \infty$, or both.

Proof. We know that $\{\mathbf{t}^k\} = \{\mathbf{p}^k + \mathbf{z}^k\}$ evolves according to $\mathbf{t}^{k+1} = G_{\lambda,A,B}(\mathbf{t}^k)$ for $A = \partial[f^* \circ (-\mathbf{M}^\mathsf{T})]$ and $B = \partial g^*$. If the set $T_\lambda{}^* = \{\ \mathbf{p} + \lambda\mathbf{z}\ \mid\ (\mathbf{p}, \mathbf{z}) \in B,\ (\mathbf{p}, -\mathbf{z}) \in A\}$ is nonempty, then $\{\mathbf{t}^k\}$ must converge to some member of it, and, following the logic of Proposition 3.42, we would have solvability of (P). Since this is contradiction, $T_\lambda{}^*$ must

be empty. It follows from Proposition 4.8 that $\{\mathbf{t}^k\}$ must be unbounded. In view of $\mathbf{t}^k = \mathbf{p}^k + \mathbf{z}^k$, at least one of $\{\mathbf{p}^k\}$ or $\{\mathbf{z}^k\}$ must be unbounded. ■

The splitting operator also allows one to construct a convergence *rate* theory for Douglas-Rachford splitting based on the convergence rate theory of the proximal point algorithm. We will see an example of this idea in Chapter 6.

It is also intriguing to consider what algorithms might be obtained by applying the *nonlinear* proximal point algorithm (Gol'shtein 1975, Gol'shtein and Tret'yakov 1979, Luque 1984b, 1986a-c) to the splitting operator. Unfortunately, such an approach does not appear to be tractable, and we will not mention it further.

Another interesting observation is that even if A and B are subdifferentials of convex functions, $S_{\lambda,A,B}$ may not be.

Proposition 4.10. There exist maximal monotone operators A and B on $\mathbb{R}^2$ and constants $\lambda > 0$ such that A and B are both subdifferentials of convex functions, but $S_{\lambda,A,B}$ is not the subdifferential of any function.

Proof. Let

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \qquad \lambda = \frac{1}{3} .$$

We will use a simplified notation in which a matrix may be used in place of the corresponding linear operator. Then

$$\mathbf{A} = \partial f, \quad \text{where} \quad f(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^{\mathsf{T}} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x}$$

$$\mathbf{B} = \partial g, \quad \text{where} \quad g(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^{\mathsf{T}} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} \quad .$$

By direct calculation,

$$J_{\lambda \mathbf{A}} = \begin{bmatrix} \frac{5}{8} & -\frac{1}{8} \\ -\frac{1}{8} & \frac{5}{8} \end{bmatrix} \qquad J_{\lambda \mathbf{B}} = \begin{bmatrix} \frac{3}{5} & 0 \\ 0 & \frac{3}{4} \end{bmatrix} \quad ,$$

hence

$$G_{\lambda,\mathbf{A},\mathbf{B}} = J_{\lambda \mathbf{A}}(2J_{\lambda \mathbf{B}} - I) + (I - J_{\lambda \mathbf{B}}) = \begin{bmatrix} \frac{1}{2} & \frac{1}{16} \\ \frac{1}{10} & \frac{11}{16} \end{bmatrix}$$

and

$$S_{\lambda,A,B} = (G_{\lambda,A,B})^{-1} - I = \begin{bmatrix} \frac{28}{27} & -\frac{5}{27} \\ -\frac{8}{27} & \frac{13}{27} \end{bmatrix} \quad .$$

Since this matrix is not symmetric, it cannot represent an operator that is the subdifferential of a function. ■

4.2. Relationship with the Method of Partial Inverses

We now introduce Spingarn' s concept of a *partial inverse* of a monotone operator, which has proved useful in the construction of decomposition algorithms. We will show that partial inverse operators are special cases of splitting operators.

Definition 4.2 (Spingarn 1983, 1985b). Let T be an operator on a Hilbert space $\mathcal{H}$, and let V be any linear subspace of $\mathcal{H}$, $V^{\perp}$ denoting its orthogonal complement. Then the *partial inverse* T_V of T with respect to V is the operator obtained by swapping the $V^{\perp}$ components of each pair in T, thus:

$$T_V = \{(x_V + y_{V^{\perp}}, y_V + x_{V^{\perp}}) \mid (x, y) \in T\} \quad .$$

Here, we use the notation that for any vector z, z_V denotes the projection of z on V, and $z_{V^\perp}$ its projection onto $V^\perp$.

Spingarn proved that for any subspace V, T_V is (maximal) monotone if and only if T is (maximal) monotone. He proposed the partial inverse as a means of solving the problem

$$\text{Find } (x, y) \in T \text{ such that } x \in V \text{ and } y \in V^\perp , \tag{ZV}$$

where T is maximal monotone. He suggested applying the proximal point algorithm to T_V to obtain $\mathbf{z}$ such that $0 = T_V \mathbf{z}$, and then projecting $\mathbf{z}$ onto V and $V^\perp$ to obtain $\mathbf{x}$ and $\mathbf{y}$, respectively. The reasoning behind this suggestion is that since $(T_V)_V = T$, as the reader may easily confirm, having $(z, 0) \in T_V$ implies that $(z_V, z_{V^\perp}) = (x, y) \in T$, while $x \in V$ and $y \in V^\perp$.

Now consider the problem

$$\text{find } x \text{ such that } (T + N_V)x \ni 0 , \tag{ZV'}$$

where $N_V = V \times V^\perp$ is the normal cone operator of V, as in Proposition 3.5. A solution x to this problem must be in dom $N_V = V$, and there must exist some $y \in Tx$ such that $-y \in V^\perp$, hence $y \in V^\perp$. Thus, (ZV') is equivalent to (ZV).

Suppose we apply the Lions-Mercier algorithm to (ZV') with $A = N_V = V \times V^\perp$, $B = T$, and $\lambda = 1$. This is equivalent to applying the proximal point algorithm to the operator $S_{1,V\times V^\perp,T}$. Now,

$$\begin{aligned} S_{1,V\times V^\perp,T} &= \{(v + b, u - v) \mid (u, b) \in T, (v, a) \in V \times V^\perp, v + a = u - b\} \\ &= \{(v + b, u - v) \mid (u, b) \in T, v \in V, a \in V^\perp, v + a = u - b\} . \end{aligned}$$

The conditions $v \in V$, $a \in V^{\perp}$, and $v + a = u - b$ are equivalent to

$$v = (u-b)_V \qquad a = (u-b)_{V^{\perp}} \ ,$$

hence

$$\begin{aligned} S_{1,V\times V^{\perp},T} &= \{((u-b)_V + b, u - (u-b)_V) \mid (u, b) \in T\} \\ &= \{(u_V + b_{V^{\perp}}, b_V + u_{V^{\perp}}) \mid (u, b) \in T\} \\ &= T_V \quad . \end{aligned}$$

Thus, we have proved:

Proposition 4.11. Let T be any maximal monotone operator on a Hilbert space $\mathcal{H}$, and let V be any linear subspace of $\mathcal{H}$. Then $S_{1,V\times V^{\perp},T} = T_V$, and applying the proximal point algorithm to these two operators produces identical results.

Although we have shown the equivalence for the case $A = V \times V^{\perp}$ and $B = T$, one gets essentially the same results when one picks $B = V \times V^{\perp}$ and $A = T$. In this case, the algebra works out slightly differently, and one obtains

$$S_{1,T,V\times V^{\perp}} = \{(x_V - y_{V^{\perp}}, x_V - y_{V^{\perp}}) \mid (x, y) \in T\} \ ,$$

which is almost identical to T_V, except for a minor difference in signs. We will sketch an analysis showing that $S_{1,T,V\times V^{\perp}}$ is essentially a representation of T_V in a different coordinate system. Let Q_V be the linear operator on $\mathcal{H}$ such that $Q_V(z) = z_V - z_{V^{\perp}}$. Then Q_V is its own inverse. As a consequence,

$$T_V = Q_V(S_{1,T,V\times V^{\perp}})Q_V \qquad \text{and} \qquad S_{1,T,V\times V^{\perp}} = Q_V(T_V)Q_V \quad .$$

Based on these observations, it is possible to show that

$$(I + T_V)^{-1} = Q_V(G_{1,T,V\times V^\perp})Q_V \qquad \text{and} \qquad G_{1,T,V\times V^\perp} = Q_V(I + T_V)^{-1}Q_V \quad .$$

Thus, the two maps $(I + T_V)^{-1}$ and $G_{1,T,V\times V^\perp}$ are coordinate transformations of one another, and applying the proximal point algorithm to $S_{1,T,V\times V^\perp}$ and T_V is equivalent, up to a change of variables.

To reiterate, the main point is that partial inverse operators are special cases of splitting operators in which one of the operators has the special form $V \times V^\perp$, where V is some linear space. This result has the interesting corollary that all algorithms based on partial inverses are in fact special cases of Douglas-Rachford splitting. One example of such an algorithm is the "progressive hedging" decomposition method for stochastic programming (Rockafellar and Wets 1987). Another is Spingarn's (1985b) decomposition method for block-separable convex programming, which we will encounter in the next chapter.

As a final topic for this section, we give an example of the use of the method of partial inverses, or equivalently Douglas-Rachford splitting, to find a zero of a sum of an arbitrary number of maximal monotone operators $T_1,...,T_S$. To describe this technique, we need to make a definition:

Definition 4.3. Let $X_1,...,X_S$ and $Y_1,...,Y_S$ be any sets sets, and let $T_1,...,T_S$ be multivalued mappings such that $T_s: X_s \rightrightarrows Y_s$ for $s = 1,...,S$. Then the *direct product* of $T_1,...,T_S$, denoted

$$T_1 \otimes T_2 \otimes ... \otimes T_S \qquad \text{or} \qquad \bigotimes_{s=1}^{S} T_s \quad ,$$

is the mapping $(X_1 \times ... \times X_S) \rightrightarrows (Y_1 \times ... \times Y_S)$ such that

$$\left(\bigotimes_{s=1}^{S} T_s \right)(x_1,...,x_S) \;=\; T_1(x_1) \times ... \times T_S(x_S) \qquad \forall\, (x_1,...,x_S) \in X_1 \times \; ... \times X_S \quad ,$$

that is,

$$\bigotimes_{s=1}^{S} T_s \;=\; \{\; ((x_1,...,x_S), (y_1,...,y_S)) \;\mid\; (x_s, y_s) \in T_s,\; s=1,...,S \;\} \quad .$$

Proposition 4.12. Suppose we have a collection of (possibly multivalued) operators $T_1,...,T_S$ on the Hilbert spaces $\mathcal{H}_1,...,\mathcal{H}_S$, respectively. Then, if $T_1,...,T_S$ are all monotone, $T_1 \otimes ... \otimes T_S$ is also monotone, where $\mathcal{H}_1 \times ... \times \mathcal{H}_S$ is endowed with the canonical inner product

$$\langle (x_1,...,x_S), (y_1,...,y_S) \rangle \;=\; \langle x_1, y_1 \rangle_1 + ... + \langle x_S, y_S \rangle_S \quad ,$$

where $\langle \cdot\,, \cdot \rangle_s$ denotes the inner product function for $\mathcal{H}_s$. Furthermore, if $T_1,...,T_S$ are all maximal, so is $T_1 \otimes ... \otimes T_S$.

Proof. Consider any $((x_1,...,x_S), (y_1,...,y_S)), ((x_1',...,x_S'), (y_1',...,y_S')) \in T_1 \otimes ... \otimes T_S$. Then $(x_s, y_s) \in T_s$, $(x_s', y_s') \in T_s$, and $\langle x_s - x_s', y_s - y_s' \rangle_s \geq 0$ for all $s = 1,...,S$. So,

$$\langle (x_1,...,x_S) - (x_1',...,x_S'), (y_1,...,y_S) - (y_1',...,y_S') \rangle \;=\; \sum_{s=1}^{S} \langle x_s - x_s', y_s - y_s' \rangle_s \;\geq\; 0 \quad .$$

So, $T_1 \otimes ... \otimes T_S$ is monotone. For maximality, we use Theorem 3.5. If $T_1,...,T_S$ are maximal, then we have $\mathrm{im}(I + T_s) = \mathcal{H}_s$ for all $s = 1,...,S$. From the definition of the direct product, we then have $\mathrm{im}(I + (T_1 \otimes ... \otimes T_S)) = \mathcal{H}_1 \times ... \times \mathcal{H}_S$, so $T_1 \otimes ... \otimes T_S$ is maximal monotone. ■

Let us now suppose we wish to find a zero of $T_1 + ... + T_S$, where $T_1,...,T_S: \mathbb{R}^n \rightrightarrows \mathbb{R}^n$. To solve this problem, Spingarn (1983) proposed applying the proximal point algorithm to T_V, where $T: (\mathbb{R}^n)^S \rightrightarrows (\mathbb{R}^n)^S$ is the operator $T_1 \otimes ... \otimes T_S$, and V is the subspace

$$V = \{(\mathbf{x}_1,...,\mathbf{x}_S) \in (\mathbb{R}^n)^S \mid \mathbf{x}_1 = \mathbf{x}_2 = ... = \mathbf{x}_S\} \quad .$$

The reasoning behind this suggestion is as follows: the orthogonal complement of V is

$$V^\perp = \{(\mathbf{y}_1,...,\mathbf{y}_S) \in (\mathbb{R}^n)^S \mid \mathbf{y}_1 + \mathbf{y}_2 + ... + \mathbf{y}_S = \mathbf{0}\} \quad ,$$

and therefore problem (ZV), for this choice of T and V, specializes to

Find $(\mathbf{x}_1, \mathbf{y}_1) \in T_1,...,(\mathbf{x}_S, \mathbf{y}_S) \in T_S$ such that $\mathbf{x}_1 = \mathbf{x}_2 = ... = \mathbf{x}_S$ and $\mathbf{y}_1 + \mathbf{y}_2 + ... + \mathbf{y}_S = \mathbf{0}$,

or equivalently,

Find $\mathbf{x}$ such that $(\mathbf{x}, \mathbf{y}_1) \in T_1,...,(\mathbf{x}, \mathbf{y}_S) \in T_S$ and $\mathbf{y}_1 + \mathbf{y}_2 + ... + \mathbf{y}_S = \mathbf{0}$,

which is the problem of finding a zero of $T_1 + ... + T_S$.

Applying the proximal point algorithm to the partial inverse T_V is, as we have seen, equivalent to applying Douglas-Rachford splitting to either $A = V \times V^\perp$ and $B = T$, or, apart from a change of variables, $A = T$ and $B = V \times V^\perp$. In the latter case, plugging $A = T$ and $B = V \times V^\perp$ into the tight Douglas-Rachford scheme of Section 3.4.4 with $\lambda = 1$ yields the algorithm

Given $\mathbf{x}^k, \mathbf{b}_1^k, \mathbf{b}_2^k,...,\mathbf{b}_S^k \in \mathbb{R}^n$, where $\mathbf{b}_1^k + \mathbf{b}_2^k + ... + \mathbf{b}_S^k = \mathbf{0}$

$$\text{Let} \quad \mathbf{y}_s^{k+1} = J_{T_s}(\mathbf{x}^k - \mathbf{b}_s^k) \qquad \text{for } s = 1,...,S$$

$$\text{Let} \quad \mathbf{x}^{k+1} = \frac{1}{S} \sum_{s=1}^{S} \mathbf{y}_s^{k+1}$$

$$\text{Let} \quad \mathbf{b}_s^{k+1} = \mathbf{y}_s^{k+1} + \mathbf{b}_s^k - \mathbf{x}^{k+1} \qquad \text{for } s = 1,...,S \quad .$$

Without giving the complete details of the derivation, we note that the calculation of the $\{\mathbf{y}_s^{k+1}\}$ corresponds to the application of the resolvent $J_A = (I+A)^{-1}$. The calculations of $\mathbf{x}^{k+1}$ and the $\{\mathbf{b}_s^{k+1}\}$ implement projections onto V and $V^{\perp}$, respectively, corresponding to the application of the resolvent $J_B = (I+B)^{-1}$. Using $A = V \times V^{\perp}$ and $B = T$ produces a method which is equivalent, up to a change of variables.

We mention this application of partial inverses because it contains the central idea used by the general monotone operator method of Gol'shtein, as we shall see in the next section.

4.3. Relationship to the Method of Gol'shtein

We now discuss the general monotone operator method of Gol'shtein (1987), upon which several convex programming decomposition methods (Gol'shtein 1985, 1986) have been based.

Gol'shtein's method is designed to solve the following problem: let $E(1),...,E(S)$ be linear subspaces of $\mathbb{R}^n$ such that $E(1),...,E(S)$ span $\mathbb{R}^n$. For any $\mathbf{z} \in \mathbb{R}^n$, let $\mathbf{z}(s)$ denote the projection $P_{E(s)}(\mathbf{z})$ of $\mathbf{z}$ onto $E(s)$. Gol'shtein actually requires that each $E(s)$ be of the form

$$E(s) = \{\mathbf{z} \mid z_j = 0 \ \forall\, j \notin P(s)\} \quad ,$$

where $P(s)$ is some proper subset of $\{1,...,n\}$. However, this restriction appears to be inessential; it only makes the $\mathbf{z}(s)$ easy to compute. Now, let $T_1,...,T_S$ be maximal monotone operators on $E(1),...,E(S)$, respectively. Note that we assert the maximality of each T_s only with respect to $E(s)$. As each T_s is a subset of $E(s) \times E(s)$, and $E(s) \subseteq \mathbb{R}^n$,

we have for all s that $T_s \subseteq \mathbb{R}^n \times \mathbb{R}^n$, and T_s may therefore be interpreted as an operator on $\mathbb{R}^n$. However, T_s cannot be maximal with respect to $\mathbb{R}^n$ unless $E(s) = \mathbb{R}^n$.

Define the operator T_Σ: $\mathbb{R}^n \rightrightarrows \mathbb{R}^n$ via

$$T_\Sigma(\mathbf{z}) = \sum_{s=1}^{S} T_s(\mathbf{z}(s)) \quad ,$$

which means T_Σ is monotone, but not necessarily maximal. Given some $\mathbf{d} \in \mathbb{R}^n$, we wish to find $\mathbf{z} \in \mathbb{R}^n$ such that $T_\Sigma(\mathbf{z}) \ni \mathbf{d}$. This problem is a slight generalization of that of finding a zero of T_Σ.

Gol'shtein's approach to solving this problems is as follows: let $\mathbf{H}_1,...,\mathbf{H}_S$ be symmetric positive semidefinite matrices such that $\text{im}(\mathbf{H}_s) = E(s)$ and the null space of $\mathbf{H}_s$ is $E(s)^\perp$ for all s. That is, each $\mathbf{H}_s$ acts like a positive definite operator on $E(s)$ and annihilates all vectors orthogonal to $E(s)$. Let $\mathbf{H} = \sum_{s=1}^{S} \mathbf{H}_s$. Gol'shtein (1987) shows that, since $E(1),...,E(S)$ span $\mathbb{R}^n$, $\mathbf{H}$ is positive definite. He further defines $\mathbf{z}^*(s, \mathbf{d}_s, \mathbf{w})$, for any $\mathbf{d}_s \in E(s)$ and $\mathbf{w} \in \mathbb{R}^n$, to be the unique solution in $\mathbf{z}$ of the inclusion

$$\mathbf{H}_s(\mathbf{z} - \mathbf{w}) + T_s\mathbf{z} \ni \mathbf{d}_s \quad ,$$

where $\mathbf{z}$ is allowed to range over $E(s)$. The reasons that $\mathbf{z}^*(s, \mathbf{d}_s, \mathbf{w})$ is unique will become apparent below. Gol'shtein proposed the following algorithm:

Choose any:

$$\mathbf{d}_1^0 \in E(1),..., \mathbf{d}_S^0 \in E(S) \qquad \text{such that} \qquad \sum_{s=1}^{S} \mathbf{d}_s^0 = \mathbf{d}$$

$$\mathbf{w}^0 \in \mathbb{R}^n$$

$\{\alpha_k\}_{k=0}^{\infty}$ such that $\sum_{k=0}^{\infty} \alpha_k < \infty$

$\{\rho_k\}_{k=0}^{\infty} \subseteq (0, 2)$ such that $0 < \inf_{k \geq 0} \rho_k \leq \sup_{k \geq 0} \rho_k < 2$.

Perform the iteration, for each $k \geq 0$,

Choose $\mathbf{z}_s^k \in E(s)$ such that $\| \mathbf{z}_s^k - \mathbf{z}^*(s, \mathbf{d}_s{}^k, \mathbf{w}^k) \| \leq \epsilon_k$ for $s = 1,...,S$

$$\mathbf{z}^k = \mathbf{H}^{-1} \sum_{s=1}^{S} \mathbf{H}_s \mathbf{z}_s^k$$

$$\mathbf{d}_s^{k+1} = \mathbf{d}_s^k - \rho_k \mathbf{H}_s (\mathbf{z}_s^k - \mathbf{z}^k)$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \rho_k (\mathbf{w}^k - \mathbf{z}^k) \quad .$$

Gol'shtein validates this method via a long, direct proof. We now proceed to derive an identical algorithm from Douglas-Rachford splitting.

Let the matrices $\mathbf{H}_1,...,\mathbf{H}_S$ be given. For each s, let $n_s = \dim[E(s)]$ and $\mathbf{B}_s$ be an orthogonal matrix whose first n_s columns are a basis for $E(s)$, and whose remaining columns are a basis for $E(s)^{\perp}$. Then

$$\mathbf{B}_s^{-1} \mathbf{H}_s \mathbf{B}_s = \begin{bmatrix} \mathbf{G}_s & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad ,$$

where $\mathbf{G}_s$ is some positive definite $n_s \times n_s$ matrix. For all s, let

$$\mathbf{Q}_s = \mathbf{H}_s^{1/2} = \mathbf{B}_s \begin{bmatrix} \mathbf{G}_s^{1/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{B}_s^{-1}$$

$$\mathbf{R}_s = \mathbf{B}_s \begin{bmatrix} \mathbf{G}_s^{-1/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{B}_s^{-1} \quad .$$

Then, for all s,

$$\mathbf{Q}_s\mathbf{R}_s = \mathbf{R}_s\mathbf{Q}_s = \mathbf{B}_s\Big[\underbrace{\begin{matrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{matrix}}_{n_s \text{ columns}}\Big]\mathbf{B}_s^{-1} \quad ,$$

that is, $\mathbf{Q}_s\mathbf{R}_s = \mathbf{R}_s\mathbf{Q}_s$ is the projection operator for $E(s)$. $\mathbf{Q}_s$ and $\mathbf{R}_s$, like $\mathbf{H}_s$, are symmetric for all s. When viewed as operators restricted to $E(s)$, $\mathbf{Q}_s$ and $\mathbf{R}_s$ are inverses.

We now rephrase Gol'shtein's definition of $\mathbf{z}^*(s, \mathbf{d}_s, \mathbf{w})$, which is supposed to be the unique root in $\mathbf{z}$ of

$$\mathbf{H}_s(\mathbf{z} - \mathbf{w}) + T_s\mathbf{z} \ni \mathbf{d}_s \quad ,$$

or equivalently,

$$\mathbf{H}_s\mathbf{z} + T_s\mathbf{z} \ni \mathbf{d}_s + \mathbf{H}_s\mathbf{w}$$

$$(\mathbf{H}_s + T_s)\mathbf{z} \ni \mathbf{d}_s + \mathbf{H}_s\mathbf{w} \quad .$$

For s=1,..., S, consider the operator $\mathbf{R}_sT_s\mathbf{R}_s$ on $E(s)$. Since $\mathbf{R}_s$ and $\mathbf{Q}_s$ act like inverses on $E(s)$, this operator can also be written

$$\mathbf{R}_sT_s\mathbf{R}_s = \{(\mathbf{Q}_s\mathbf{x}, \mathbf{R}_s\mathbf{y}) \mid (\mathbf{x}, \mathbf{y}) \in T_s\} \quad .$$

By logic similar to Proposition 3.1(iv), but working in the space $E(s)$, rather than $\mathbb{R}^n$, we obtain that $\mathbf{R}_sT_s\mathbf{R}_s = (\mathbf{R}_s)^{\mathsf{T}}T_s\mathbf{R}_s$ is maximal monotone on $E(s)$. Let $\mathbf{z}^*$ be a member of $E(s)$. Using the distributive laws of Lemma 3.3 (in Section 3.5.1),

$$\begin{aligned}
& (\mathbf{H}_s + T_s)\mathbf{z}^* \ni \mathbf{d}_s + \mathbf{H}_s\mathbf{w} \\
\Leftrightarrow \quad & \mathbf{R}_s(\mathbf{H}_s + T_s)\mathbf{z}^* \ni \mathbf{R}_s\mathbf{d}_s + \mathbf{R}_s\mathbf{H}_s\mathbf{w} = \mathbf{R}_s\mathbf{d}_s + \mathbf{Q}_s\mathbf{w} \\
\Leftrightarrow \quad & (\mathbf{Q}_s + \mathbf{R}_sT_s)\mathbf{z}^* \ni \mathbf{R}_s\mathbf{d}_s + \mathbf{Q}_s\mathbf{w} \\
\Leftrightarrow \quad & (\mathbf{Q}_s + \mathbf{R}_sT_s)\mathbf{R}_s\mathbf{Q}_s\mathbf{z}^* \ni \mathbf{R}_s\mathbf{d}_s + \mathbf{Q}_s\mathbf{w} \\
\Leftrightarrow \quad & (\mathbf{I} + \mathbf{R}_sT_s\mathbf{R}_s)(\mathbf{Q}_s\mathbf{z}^*) \ni \mathbf{R}_s\mathbf{d}_s + \mathbf{Q}_s\mathbf{w}
\end{aligned}$$

$$\Leftrightarrow \quad \mathbf{Q}_s\mathbf{z}^* = (\mathbf{I} + \mathbf{R}_s T_s \mathbf{R}_s)^{-1}(\mathbf{R}_s\mathbf{d}_s + \mathbf{Q}_s\mathbf{w})$$

$$\Leftrightarrow \quad \mathbf{z}^* = \mathbf{R}_s(\mathbf{I} + \mathbf{R}_s T_s \mathbf{R}_s)^{-1}(\mathbf{R}_s\mathbf{d}_s + \mathbf{Q}_s\mathbf{w}) \quad .$$

By the maximal monotonicity of $\mathbf{R}_s T_s \mathbf{R}_s$, we obtain the existence and uniqueness of $\mathbf{z}^* = \mathbf{z}^*(s, \mathbf{d}_s, \mathbf{w})$.

These preliminaries aside, we now claim that Gol'shtein's method is a special case of Douglas-Rachford splitting with under/over-relaxation and approximate calculation, as in Propositions 4.5 and 4.6. The two operators used in the splitting are

$$A, B: \ [E(1) \times \ldots \times E(S)] \ \rightrightarrows \ [E(1) \times \ldots \times E(S)] \quad ,$$

where

$$A = \bigotimes_{s=1}^{S} \mathbf{R}_s T_s \mathbf{R}_s = (\mathbf{R}_1 T_1 \mathbf{R}_1) \otimes (\mathbf{R}_2 T_2 \mathbf{R}_2) \otimes \ldots \otimes (\mathbf{R}_S T_S \mathbf{R}_S)$$

$$B = \{((\mathbf{u}_1,\ldots,\mathbf{u}_S), (\mathbf{v}_1,\ldots,\mathbf{v}_S)) \mid \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s = -\mathbf{d}, \exists\, \mathbf{w} \in \mathbb{R}^n: \mathbf{u}_s = \mathbf{Q}_s\mathbf{w} \ \forall s \} \quad .$$

For the remainder of this section, we will use $\mathcal{H}$ to denote the space $E(1) \times \ldots \times E(S)$.

Lemma 4.1. *A and B as defined above are maximal monotone on $\mathcal{H} = E(1) \times \ldots \times E(S)$.*

Proof. We have already argued that each operator $\mathbf{R}_s T_s \mathbf{R}_s$ is maximal monotone on $E(s)$. Proposition 4.12 therefore implies the maximal monotonicity of A. We now turn to B. Since $E(1),\ldots,E(S)$ span $\mathbb{R}^n$, there exist $\mathbf{c}_1 \in E(1),\ldots, \mathbf{c}_S \in E(S)$ such that $\mathbf{c}_1 + \ldots + \mathbf{c}_S = -\mathbf{d}$. Letting $\mathbf{b}_s = \mathbf{R}_s\mathbf{c}_s$ for all s, we have $\mathbf{b} = (\mathbf{b}_1,\ldots, \mathbf{b}_S) \in \mathcal{H}$ such that $\sum_{s=1}^{S} \mathbf{Q}_s\mathbf{b}_s = -\mathbf{d}$. Let $U \subseteq \mathcal{H}$ be the linear subspace

$$U = \{(\mathbf{u}_1,\ldots, \mathbf{u}_S) \in \mathcal{H} \mid \exists\, \mathbf{w} \in \mathbb{R}^n: \mathbf{u}_s = \mathbf{Q}_s\mathbf{w} \ \forall\, s = 1,\ldots,S\} \quad .$$

The reader may verify that the orthogonal complement of U relative to $\mathcal{H}$ is

$$U^{\perp} = \{(\mathbf{v}_1,..., \mathbf{v}_S) \in \mathcal{H} \mid \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s = \mathbf{0}\} \quad .$$

Therefore B is of the form $U \times (U^{\perp} + \mathbf{b})$, and so, by Proposition 3.5, $B = (N_{(U^{\perp} + \mathbf{b})})^{-1}$, where $N_{(U^{\perp} + \mathbf{b})}$ denotes the normal cone operator of $U^{\perp} + \mathbf{b}$. This operator is maximal monotone, so B is maximal monotone. ■

We now show that locating a zero $A+B$ is essentially equivalent to solving $T_{\Sigma}(\mathbf{z}) \ni \mathbf{d}$.

Proposition 4.13. All zeroes of $A+B$ must be of the form $(\mathbf{Q}_1\mathbf{w}^*,\mathbf{Q}_2\mathbf{w}^*,..., \mathbf{Q}_S\mathbf{w}^*)$ for some $\mathbf{w}^* \in \mathbb{R}^n$, and

$$(A+B)(\mathbf{Q}_1\mathbf{w}^*,..., \mathbf{Q}_S\mathbf{w}^*) \ni \mathbf{0} \iff T_{\Sigma}(\mathbf{w}^*) \ni \mathbf{d} \quad .$$

Proof. To be a zero of $A+B$, $(\mathbf{u}_1,..., \mathbf{u}_S)$ must at the very least be a member of dom B. Therefore, $\mathbf{u}_s = \mathbf{Q}_s\mathbf{w}^*$ for some $\mathbf{w}^* \in \mathbb{R}^n$. This proves the first assertion. Now,

$$\begin{aligned}
& (A+B)(\mathbf{Q}_1\mathbf{w}^*,..., \mathbf{Q}_S\mathbf{w}^*) \ni \mathbf{0} \\
\Leftrightarrow \quad & \exists\, (\mathbf{v}_1^*,..., \mathbf{v}_S^*) \in \mathcal{H}\colon \ \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s^* = -\mathbf{d}, \quad \mathbf{R}_sT_s\mathbf{R}_s(\mathbf{Q}_s\mathbf{w}^*) \ni -\mathbf{v}_s^* \ \forall\, s \\
\Leftrightarrow \quad & \exists\, (\mathbf{v}_1^*,..., \mathbf{v}_S^*) \in \mathcal{H}\colon \ \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s^* = -\mathbf{d}, \quad \mathbf{R}_sT_s\mathbf{w}^*(s) \ni -\mathbf{v}_s^* \ \forall\, s \\
\Leftrightarrow \quad & \exists\, (\mathbf{v}_1^*,..., \mathbf{v}_S^*) \in \mathcal{H}\colon \ \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s^* = -\mathbf{d}, \quad T_s\mathbf{w}^*(s) \ni -\mathbf{Q}_s\mathbf{v}_s^* \ \forall\, s \quad .
\end{aligned}$$

If this last condition holds,

$$T_{\Sigma}(\mathbf{w}^*) = \sum_{s=1}^{S} T_s\mathbf{w}^*(s) \ni \sum_{s=1}^{S} (-\mathbf{Q}_s\mathbf{v}_s^*) = -(-\mathbf{d}) = \mathbf{d} \quad .$$

Conversely, given $\mathbf{z}^*$ such that $T_\Sigma(\mathbf{z}^*) \ni \mathbf{d}$, we know there exist $\mathbf{d}_1 \in T_1\mathbf{w}^*(1),...,$ $\mathbf{d}_S \in T_S\mathbf{w}^*(S)$ such that $\sum_{s=1}^{S} \mathbf{d}_s = \mathbf{d}$. Letting $\mathbf{v}_s^* = -\mathbf{R}_s\mathbf{d}_s$ for all s, one obtains the condition

$$\exists\, (\mathbf{v}_1^*,..., \mathbf{v}_S^*) \in \mathcal{H}\colon\ \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s^* = -\mathbf{d},\quad T_s\mathbf{w}^*(s) \ni -\mathbf{Q}_s\mathbf{v}_s^*\ \forall\, s\ ,$$

equivalent to $(A+B)(\mathbf{Q}_1\mathbf{w}^*,..., \mathbf{Q}_S\mathbf{w}^*) \ni \mathbf{0}$. ■

We now apply Proposition 4.6 to A and B to obtain an algorithm equivalent to Gol'shtein's. Rephrasing the method of Proposition 4.6, we wish to create an algorithm of the form

(1) Given $(\mathbf{u}^k, \mathbf{b}^k) \in \mathcal{H}\times\mathcal{H}$, let $(\mathbf{v}^{k+1}, \mathbf{a}^{k+1}) \in \mathcal{H}\times\mathcal{H}$ be a pair within α_k of being in A such that $\mathbf{v}^{k+1} + \lambda\mathbf{a}^{k+1} = \mathbf{u}^k - \lambda\mathbf{b}^k$.

(2) Let $(\mathbf{u}^{k+1}, \mathbf{b}^{k+1}) \in \mathcal{H}\times\mathcal{H}$ be a pair within β_k of being in B such that $\mathbf{u}^{k+1} + \lambda\mathbf{b}^{k+1} = \rho_k\mathbf{v}^{k+1} + (1-\rho_k)\mathbf{u}^k + \lambda\mathbf{b}^k$.

By a pair $(\mathbf{v}, \mathbf{a})$ being "within α of being in A", we mean that $\|\mathbf{v} - J_{\lambda A}(\mathbf{v} + \lambda\mathbf{a})\| \le \alpha$; a similar interpretation is intended in (2).

From now on, we will fix λ at 1. Our first concern is implementing the computation (1). Let us suppose we are given $\mathbf{u}^k = (\mathbf{u}_1{}^k,..., \mathbf{u}_S{}^k) = (\mathbf{Q}_1\mathbf{w}^k,..., \mathbf{Q}_S\mathbf{w}^k)$ and $\mathbf{b}^k = (\mathbf{b}_1{}^k,..., \mathbf{b}_S{}^k)$ such that $\sum_{s=1}^{S}\mathbf{Q}_s\mathbf{b}_s{}^k = -\mathbf{d}$. To implement (1), we may calculate for $s = 1,..., S$, some $\mathbf{v}_s{}^{k+1}$ such that

$$\|\mathbf{v}_s{}^{k+1} - (\mathbf{I} + \mathbf{R}_sT_s\mathbf{R}_s)^{-1}(\mathbf{Q}_s\mathbf{w}^k - \mathbf{b}_s)\| \le \frac{\alpha_k}{\sqrt{S}}\ ,$$

so that, setting $\mathbf{v}^{k+1} = (\mathbf{v}_1{}^{k+1},..., \mathbf{v}_S{}^{k+1})$, we have

$$\| \mathbf{v}_s^{k+1} - J_A(\mathbf{u}^k - \mathbf{b}^k) \| \leq \sqrt{S}\left(\frac{\alpha_k}{\sqrt{S}}\right) = \alpha_k \ .$$

We now implement step (2) in its exact, $\beta_k = 0$ form, that is

$$\mathbf{u}^{k+1} = J_B(\rho_k \mathbf{v}^{k+1} + (1 - \rho_k)\mathbf{u}^k + \mathbf{b}^k) \ .$$

To do this, we need to understand how to evaluate the operator J_B.

Lemma 4.2. Given any $\mathbf{y} = (\mathbf{y}_1, ..., \mathbf{y}_S) \in \mathcal{H}$,

$$J_B(\mathbf{y}_1, ..., \mathbf{y}_S) = (\mathbf{Q}_1\mathbf{z}, ..., \mathbf{Q}_S\mathbf{z}) \ ,$$

where

$$\mathbf{z} = \mathbf{H}^{-1}\left[\sum_{s=1}^{S} \mathbf{Q}_s\mathbf{y}_s + \mathbf{d} \right] \ .$$

Proof. By construction, $(\mathbf{Q}_1\mathbf{z}, ..., \mathbf{Q}_S\mathbf{z}) \in \text{dom } B$. We need only verify that

$$(\mathbf{r}_1, ..., \mathbf{r}_S) = (\mathbf{y}_1, ..., \mathbf{y}_S) - (\mathbf{Q}_1\mathbf{z}, ..., \mathbf{Q}_S\mathbf{z})$$

has the property that $(\mathbf{r}_1, ..., \mathbf{r}_S) \in B(\mathbf{Q}_1\mathbf{z}, ..., \mathbf{Q}_S\mathbf{z})$, that is, that $\sum_{s=1}^{S} \mathbf{Q}_s\mathbf{r}_s = -\mathbf{d}$. Now, $\mathbf{z} = \mathbf{H}^{-1}\left[\sum_{s=1}^{S} \mathbf{Q}_s\mathbf{y}_s + \mathbf{d} \right]$ implies that $\sum_{s=1}^{S} \mathbf{Q}_s\mathbf{y}_s = \mathbf{H}\mathbf{z} - \mathbf{d}$. Thus,

$$\begin{aligned}
\sum_{s=1}^{S} \mathbf{Q}_s\mathbf{r}_s &= \sum_{s=1}^{S} \mathbf{Q}_s(\mathbf{y}_s - \mathbf{Q}_s\mathbf{z}) \\
&= \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{y}_s - \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{Q}_s\mathbf{z} \\
&= (\mathbf{H}\mathbf{z} - \mathbf{d}) - \sum_{s=1}^{S} \mathbf{H}_s\mathbf{z} \\
&= \mathbf{H}\mathbf{z} - \mathbf{d} - \left(\sum_{s=1}^{S} \mathbf{H}_s \right)\mathbf{z}
\end{aligned}$$

$$= \mathbf{Hz} - \mathbf{d} - \mathbf{Hz}$$

$$= -\mathbf{d} \quad . \quad \blacksquare$$

Thus, we can calculate $\mathbf{u}^{k+1}$ by

$$\begin{aligned}
\mathbf{w}^{k+1} &= \mathbf{H}^{-1}\left[\sum_{s=1}^{S} \mathbf{Q}_s\left(\rho_k \mathbf{v}_s{}^{k+1} + (1-\rho_k)\mathbf{Q}_s\mathbf{w}^k + \mathbf{b}_s{}^k\right) + \mathbf{d}\right] \\
&= \mathbf{H}^{-1}\left[\rho_k \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s{}^{k+1} + (1-\rho_k)\sum_{s=1}^{S} \mathbf{H}_s\mathbf{w}^k + \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{b}_s{}^k + \mathbf{d}\right] \\
&= \mathbf{H}^{-1}\left[\rho_k \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s{}^{k+1} + (1-\rho_k)\sum_{s=1}^{S} \mathbf{H}_s\mathbf{w}^k - \mathbf{d} + \mathbf{d}\right] \\
&= \mathbf{H}^{-1}\left[\rho_k \sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s{}^{k+1} + (1-\rho_k)\sum_{s=1}^{S} \mathbf{H}_s\mathbf{w}^k\right] \\
&= \rho_k\mathbf{H}^{-1}\sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s{}^{k+1} + (1-\rho_k)\mathbf{w}^k \quad ,
\end{aligned}$$

followed by $\mathbf{u}_s{}^{k+1} = \mathbf{Q}_s\mathbf{w}^{k+1}$ for $s = 1,...,S$. Notice that

$$\mathbf{w}^{k+1} = \rho_k\mathbf{H}^{-1}\sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s{}^{k+1} + (1-\rho_k)\mathbf{w}^k$$

can be rearranged into

$$\mathbf{w}^k - \mathbf{w}^{k+1} = \rho_k(\mathbf{w}^k - \mathbf{H}^{-1}\sum_{s=1}^{S} \mathbf{Q}_s\mathbf{v}_s{}^{k+1}) \quad .$$

To find $\mathbf{b}^{k+1}$, we now solve

$$\mathbf{u}^{k+1} + \mathbf{b}^{k+1} = \rho_k\mathbf{v}^{k+1} + (1-\rho_k)\mathbf{u}^k + \mathbf{b}^k$$

for $\mathbf{b}^{k+1}$, getting for each s that

$$
\begin{aligned}
\mathbf{b}_s^{k+1} &= \rho_k \mathbf{v}_s^{k+1} + (1-\rho_k)\mathbf{u}_s^k + \mathbf{b}_s^k - \mathbf{u}_s^{k+1} \\
&= \mathbf{b}_s^k + \rho_k \mathbf{v}_s^{k+1} + (1-\rho_k)\mathbf{Q}_s\mathbf{w}^k - \mathbf{Q}_s\mathbf{w}^{k+1} \\
&= \mathbf{b}_s^k + \rho_k(\mathbf{v}_s^{k+1} - \mathbf{Q}_s\mathbf{w}^k) + \mathbf{Q}_s(\mathbf{w}^k - \mathbf{w}^{k+1}) \\
&= \mathbf{b}_s^k + \rho_k(\mathbf{v}_s^{k+1} - \mathbf{Q}_s\mathbf{w}^k) + \rho_k\mathbf{Q}_s(\mathbf{w}^k - \mathbf{H}^{-1}\sum_{s=1}^{S}\mathbf{Q}_s\mathbf{v}_s^{k+1}) \\
&= \mathbf{b}_s^k + \rho_k(\mathbf{v}_s^{k+1} - \mathbf{Q}_s\mathbf{H}^{-1}\sum_{s=1}^{S}\mathbf{Q}_s\mathbf{v}_s^{k+1}) \quad .
\end{aligned}
$$

The entire algorithm may then be written

$$
\begin{aligned}
&\text{Choose } \mathbf{v}_s^{k+1} : \|\mathbf{v}_s^{k+1} - (\mathbf{I} + \mathbf{R}_s T_s \mathbf{R}_s)^{-1}(\mathbf{Q}_s\mathbf{w}^k - \mathbf{b}_s^k)\| \le \frac{\alpha_k}{\sqrt{S}}, \quad s = 1,...S \\
&\mathbf{w}^{k+1} = \rho_k\mathbf{H}^{-1}\sum_{s=1}^{S}\mathbf{Q}_s\mathbf{v}_s^{k+1} + (1-\rho_k)\mathbf{w}^k \\
&\mathbf{b}_s^{k+1} = \mathbf{b}_s^k + \rho_k(\mathbf{v}_s^{k+1} - \mathbf{Q}_s\mathbf{H}^{-1}\sum_{s=1}^{S}\mathbf{Q}_s\mathbf{v}_s^{k+1}) , \qquad s = 1,...S \quad .
\end{aligned}
$$

For all s and k, we let $\mathbf{z}_s^k = \mathbf{R}_s\mathbf{v}_s^{k+1}$ and $\mathbf{d}_s^k = -\mathbf{Q}_s\mathbf{b}_s^k$. Rewriting in terms of these new variables, the above algorithm may be implemented via

$$
\begin{aligned}
&\text{Choose } \mathbf{z}_s^k : \|\mathbf{z}_s^k - \mathbf{R}_s(\mathbf{I} + \mathbf{R}_s T_s \mathbf{R}_s)^{-1}(\mathbf{Q}_s\mathbf{w}^k + \mathbf{R}_s\mathbf{d}_s^k)\| \le \frac{\alpha_k}{\sqrt{S}\,\|\mathbf{Q}_s\|}, \quad s = 1,...S \\
&\mathbf{w}^{k+1} = \rho_k\mathbf{H}^{-1}\sum_{s=1}^{S}\mathbf{H}_s\mathbf{z}_s^k + (1-\rho_k)\mathbf{w}^k \\
&\mathbf{d}_s^{k+1} = \mathbf{d}_s^k - \rho_k(\mathbf{H}_s\mathbf{z}_s^k - \mathbf{H}_s\mathbf{H}^{-1}\sum_{s=1}^{S}\mathbf{H}_s\mathbf{z}_s^k) , \qquad s = 1,...S \quad .
\end{aligned}
$$

Further rewriting and using that

$$
\mathbf{z}^*(s, \mathbf{d}_s, \mathbf{w}) = \mathbf{R}_s(\mathbf{I} + \mathbf{R}_s T_s \mathbf{R}_s)^{-1}(\mathbf{R}_s\mathbf{d}_s + \mathbf{Q}_s\mathbf{w}) \quad ,
$$

we have the algorithm

$$\text{Choose } \mathbf{z}_s^k : \ \|\mathbf{z}_s^k - \mathbf{z}^*(s, \mathbf{d}_s, \mathbf{w})\| \ \leq \ \frac{\alpha_k}{\sqrt{S}\,\|\mathbf{Q}_s\|}, \qquad s = 1,...S$$

$$\mathbf{z}^k \ = \ \mathbf{H}^{-1} \sum_{s=1}^{S} \mathbf{H}_s \mathbf{z}_s^k$$

$$\mathbf{w}^{k+1} \ = \ \rho_k \mathbf{z}^k \ + \ (1 - \rho_k)\mathbf{w}^k$$

$$\mathbf{d}_s^{k+1} \ = \ \mathbf{d}_s^k \ - \ \rho_k(\mathbf{H}_s \mathbf{z}_s^k - \ \mathbf{H}_s \mathbf{z}^k) \ , \qquad s = 1,...S \quad .$$

Let $\alpha_k = \sqrt{S}\ \max_s\{\|\mathbf{Q}_s\|\}\varepsilon_k$ for all k, which guarantees that $\varepsilon_k \leq \frac{\alpha_k}{\sqrt{S}\,\|\mathbf{Q}_s\|}$ for all s and k, and that $\{\alpha_k\}$ is summable. We can now implement the algorithm, with a little more rewriting, as

$$\text{Choose } \mathbf{z}_s^k : \ \|\mathbf{z}_s^k - \mathbf{z}^*(s, \mathbf{d}_s, \mathbf{w})\| \ \leq \ \varepsilon_k \quad , \qquad s = 1,...S$$

$$\mathbf{z}^k \ = \ \mathbf{H}^{-1} \sum_{s=1}^{S} \mathbf{H}_s \mathbf{z}_s^k$$

$$\mathbf{w}^{k+1} \ = \ \mathbf{w}^k \ - \ \rho_k(\mathbf{w}^k - \mathbf{z}^k)$$

$$\mathbf{d}_s^{k+1} \ = \ \mathbf{d}_s^k \ - \ \rho_k \mathbf{H}_s(\mathbf{z}_s^k - \ \mathbf{z}^k) \ , \qquad s = 1,...S \quad ,$$

which is Gol'shtein's method. We have now proven

Proposition 4.14. Gol'shtein algorithm for solving $T_\Sigma(\mathbf{z}) \ni \mathbf{d}$ is one possible implementation of the generalized Douglas-Rachford splitting scheme of Proposition 4.6, as applied to the operators

$$A \ = \ \bigotimes_{s=1}^{S} \mathbf{R}_s T_s \mathbf{R}_s \ = \ (\mathbf{R}_1 T_1 \mathbf{R}_1) \otimes (\mathbf{R}_2 T_2 \mathbf{R}_2) \otimes ... \otimes (\mathbf{R}_S T_S \mathbf{R}_S)$$

and

$$B \ = \ \{((\mathbf{u}_1,...,\mathbf{u}_S), (\mathbf{v}_1,...,\mathbf{v}_S)) \ | \ \sum_{s=1}^{S} \mathbf{Q}_s \mathbf{v}_s = -\,\mathbf{d}, \exists\, \mathbf{z} \in \mathbb{R}^l : \mathbf{u}_s = \mathbf{Q}_s \mathbf{z}\ \forall s \ \} \quad .$$

This result has the immediate consequence that the convex programming decomposition methods Gol'shtein (1985, 1986) has based on his $T_\Sigma(\mathbf{z}) \ni \mathbf{d}$ method are necessarily special cases of Douglas-Rachford splitting.

One might also consider deriving an even more general $T_\Sigma(\mathbf{z}) \ni \mathbf{d}$ method by dropping the restriction, made above, that $\lambda = 1$. However, introducing a stepsize λ other than 1 turns out to yield the same method that one gets by multiplying all the matrices $\mathbf{H}_s$ by the scalar $\frac{1}{\lambda}$. Thus, no real generality can be gained in this manner.

Also, note that in the special case of Gol'shtein's problem in which $E(s) = \mathbb{R}^n$ and $\mathbf{H}_s = \mathbf{I}$ for all s, and $\mathbf{d} = \mathbf{0}$, we have

$$A = \bigotimes_{s=1}^{S} T_s \qquad \text{and} \qquad B = N_V \ ,$$

where

$$V = \{(\mathbf{x}_1,...,\mathbf{x}_S) \in (\mathbb{R}^n)^S \mid \mathbf{x}_1 = \mathbf{x}_2 = ... = \mathbf{x}_S\} \ .$$

If we further set $\rho_k = 1$ and $\varepsilon_k = 0$ for all k, Gol'shtein's method reduces to Spingarn's 1983 method for finding a zero of a sum of S operators, as described in the last section. Thus, while Gol'shtein's methods extends Spingarn's, the two algorithms are very closely related.

Finally, Figure 13 summarizes the relationships between the various algorithms that we have discussed so far.

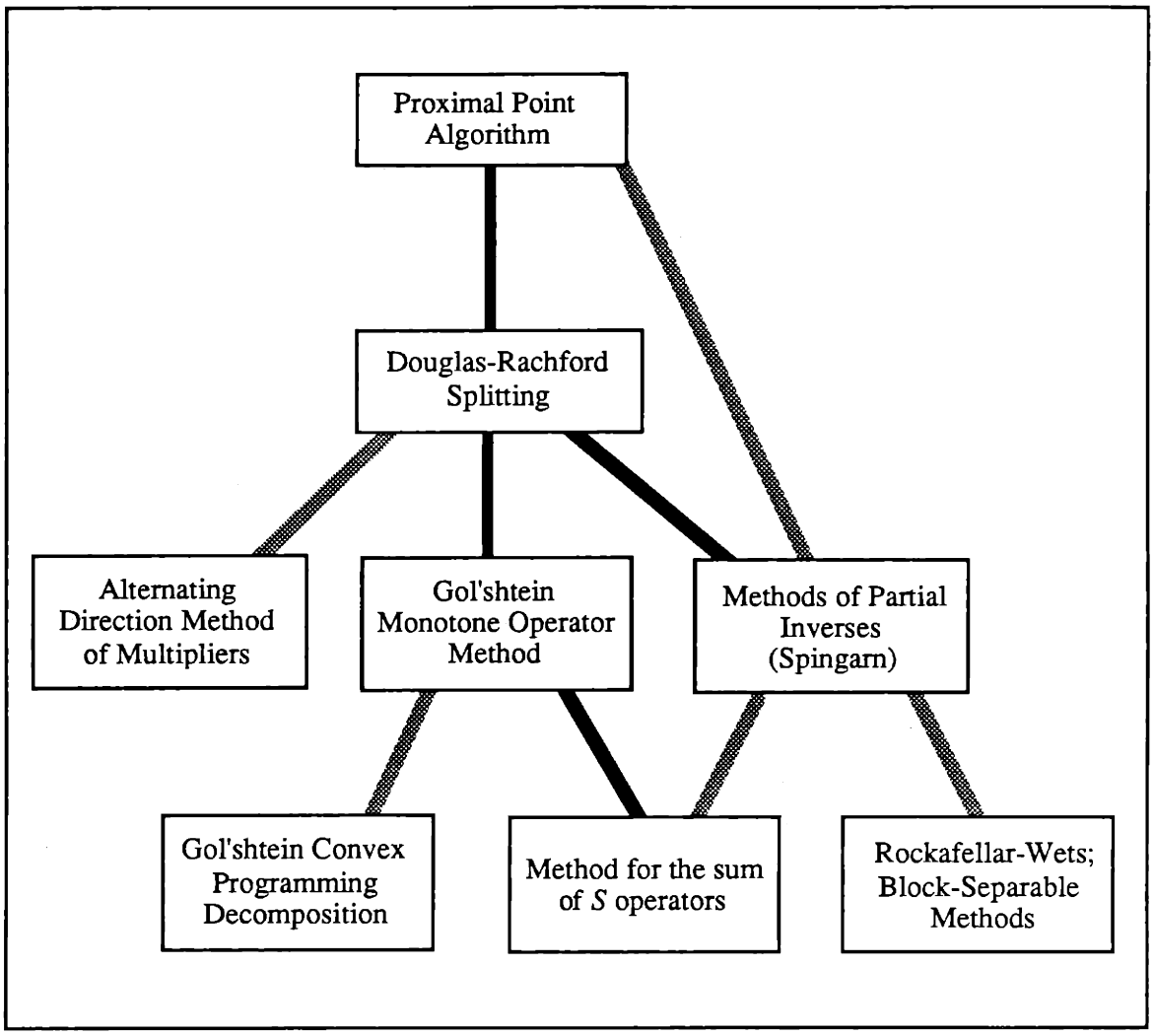


Figure 13: Taxonomy of algorithms discussed. Black lines indicate relationships established here or in Eckstein (1988), and gray lines indicate relationships that were already known.

Chapter 5

Decomposition Methods for Convex Programming

This chapter will present various ways in which operator splitting may be applied to convex programs with special structure, yielding decomposition methods. These methods are meant to be amenable to parallel implementation. The basic idea is to choose operators A and B such that a zero of $A+B$ is a solution of the convex program, but that A and B individually have a comparatively simple structure. We will focus on applications of the alternating direction method of multipliers, because of the superior general convergence properties it inherits from Douglas-Rachford splitting. However, other solution schemes are applicable to the operator splittings that follow, under suitable restrictive assumptions.

We will discuss four main algorithms, intended for problems with successively more specialized structure, and each based on a different strategy for choosing A and B. The first is a simple decomposition method for multiple set constraints, paralleling the work of Han and Lou (1988) mentioned in Section 3.5.4, but replacing the forward-backward solution scheme with the Douglas-Rachford scheme. The second topic is Spingarn's (1985b) decomposition method for block-separable convex programming, and the third is a new alternative to that algorithm, called the *epigraphic projection method.* Finally, as a bridge to the following chapter on linear programming, we will present the *alternating step method* for monotropic programming.

All the algorithms presented here, except Spingarn's block-separable method, are original. We present Spingarn's approach in order to contrast it with epigraphic projection, to which it is related. Also, a version of the alternating step method for monotropic programming has already appeared in Bertsekas and Tsitsiklis (1989), but as a result of joint work with the author.

There are certainly other applications of splitting to convex programming. For further examples, the reader should refer to Bertsekas and Tsitsiklis (1989), which develops additional special cases of the alternating direction method of multipliers, and to Gol'shtein (1985, 1986). Bertsekas and Tsitsiklis also give applications to variational inequalities, which are not covered in this thesis.

Throughout this chapter, we will construct splittings in the same way. In each section, we first find a matrix $\mathbf{M}$ and functions f and g such that the original structured convex program is equivalent to the problem (P) of Section 3.5,

$$\text{minimize}_{\ \mathbf{x}\in\mathbb{R}^n}\ f(\mathbf{x}) + g(\mathbf{M}\mathbf{x})\ , \tag{P}$$

and then apply the alternating direction method of multipliers to (P). The alternating direction method of multipliers, as demonstrated in Section 3.5.6, is equivalent to applying Douglas-Rachford splitting to the operators $A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$.

5.1. Problems with Multiple Set Constraints

In this section we consider convex programs with the structure discussed in Section 3.5.4,

$$\begin{array}{ll} \text{minimize} & h(\mathbf{x}) \\ \text{subject to} & \mathbf{x} \in C_1 \cap \ldots \cap C_s\,, \end{array} \tag{HLP}$$

where $h: \mathbb{R}^n \to (-\infty, +\infty]$ is closed proper convex and $C_1,...,C_s \subseteq \mathbb{R}^n$ are closed convex sets. The principal assumption is that the sets $C_1,...,C_s$ are simple enough that projection onto any individual C_i is a relatively easy operation, but that projection onto the intersection $C_1 \cap ... \cap C_s$ may not be. We propose a method, similar to that of Han and Lou (1988), for taking advantage of such special structure.

We convert (HLP) to the form (P) in same manner as in Section 3.5.4. That is, we take $\mathbf{M}$ to be the $ns \times n$ matrix

$$\mathbf{M} = \left.\begin{bmatrix} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{bmatrix}\right\} s \text{ times} \quad .$$

We let $f = h$, and let $g: \mathbb{R}^{ns} \to (-\infty, \infty]$ be the function $\delta_{(C_1 \times ... \times C_s)}$, that is, $g(\mathbf{z}) = 0$ if $\mathbf{z} \in C_1 \times ... \times C_s$, otherwise $g(\mathbf{z}) = \infty$. Now, $g(\mathbf{Mx})$ is zero if $\mathbf{x}$ is a member of all of the sets $C_1,...,C_s$, and is $+\infty$ otherwise; therefore $g \circ \mathbf{M} = \delta_{(C_1 \cap ... \cap C_s)}$ and (P) can be rewritten

$$\text{minimize}_{\mathbf{x} \in \mathbb{R}^n} \; h(\mathbf{x}) + \delta_{(C_1 \cap ... \cap C_s)}(\mathbf{x}) \quad ,$$

which is clearly equivalent to (HLP). Furthermore, $\mathbf{M}$, as defined above, has full column rank.

As has already been claimed, applying the dual forward-backward method to the above choices of f, g, and $\mathbf{M}$ yields the algorithm proposed by Han and Lou (1988). Instead, we propose to apply the alternating direction method of multipliers with a fixed value of $\lambda > 0$,

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x}\rangle + \tfrac{\lambda}{2} \|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2\}$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z}\rangle + \tfrac{\lambda}{2} \|\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}\|^2\} \qquad (\text{ADMOM}_\lambda)$$

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) \quad .$$

Let

$$\mathbf{p}^k = [\, p_{ij}{}^k] \in \mathbb{R}^{mn},\ 1 \le i \le m,\ 1 \le j \le n \ ,$$

denote the vector of multipliers at step k, and $\mathbf{p}_i{}^k$ denote the subvector with elements $p_{ij}{}^k$, $1 \le j \le n$. A similar indexing scheme applies to $\mathbf{z}^k \in \mathbb{R}^{mn}$. Taking advantage of the special form of f, g, and $\mathbf{M}$, we now rewrite the method (ADMOM$_\lambda$). For the first step (the minimization over $\mathbf{x}$), note that $\langle \mathbf{p}^k, \mathbf{M}\mathbf{x}\rangle = \langle \mathbf{M}^T\mathbf{p}^k, \mathbf{x}\rangle$, and $\mathbf{M}^T\mathbf{p}^k = \sum_{i=1}^{s} \mathbf{p}_i{}^k$. Also,

$$\begin{aligned}
\|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2 &= \sum_{i=1}^{s} \|\mathbf{x} - \mathbf{z}_i{}^k\|^2 \\
&= \sum_{i=1}^{s} (\|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \mathbf{z}_i{}^k\rangle + \|\mathbf{z}_i{}^k\|^2) \\
&= s\|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \sum_{i=1}^{s} \mathbf{z}_i{}^k\rangle + \sum_{i=1}^{s} \|\mathbf{z}_i{}^k\|^2 \\
&= s(\|\mathbf{x}\|^2 - 2\langle \mathbf{x}, \tfrac{1}{s}\sum_{i=1}^{s} \mathbf{z}_i{}^k\rangle + \tfrac{1}{s}\sum_{i=1}^{s} \|\mathbf{z}_i{}^k\|^2) \quad .
\end{aligned}$$

Disregarding a constant term in the minimand, we then find that $\mathbf{x}^{k+1}$ can be obtained by

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \{h(\mathbf{x}) + \langle \sum_{i=1}^{s} \mathbf{p}_i{}^k, \mathbf{x}\rangle + \tfrac{s\lambda}{2} \|\mathbf{x} - \tfrac{1}{s}\sum_{i=1}^{s} \mathbf{z}_i{}^k\|^2\} \quad .$$

The subsequent calculation of $\mathbf{z}^{k+1}$,

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z}\rangle + \tfrac{\lambda}{2} \|\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}\|^2\} \ ,$$

can be rewritten, by completing the square, as

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{ g(\mathbf{z}) + \frac{\lambda}{2} \| \mathbf{z} - (\mathbf{M}\mathbf{x}^{k+1} + \frac{1}{\lambda}\mathbf{p}^k) \|^2 \} \quad .$$

Using the special form of g and $\mathbf{M}$, this calculation decomposes by subvector $\mathbf{z}_i^{k+1}$, and may be expressed as

$$\mathbf{z}_i^{k+1} = P_{C_i}[\mathbf{x}^{k+1} - \frac{1}{\lambda}\mathbf{p}_i^k] \quad , \qquad i = 1,...,s \quad .$$

By using the special form of $\mathbf{M}$ in rewriting the multiplier update, the entire alternating direction method of multipliers now becomes

$$\begin{aligned}
&\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \{ h(\mathbf{x}) + \langle \sum_{i=1}^{s} \mathbf{p}_i^k, \mathbf{x} \rangle + \frac{s\lambda}{2} \| \mathbf{x} - \frac{1}{s}\sum_{i=1}^{s} \mathbf{z}_i^k \|^2 \} \\
&\mathbf{z}_i^{k+1} = P_{C_i}[\mathbf{x}^{k+1} - \frac{1}{\lambda}\mathbf{p}_i^k] \qquad\qquad i = 1,...,s \qquad\qquad \text{(MSM)} \\
&p_{ij}^{k+1} = p_{ij}^k + \lambda(x_j^{k+1} - z_{ij}^{k+1}) \qquad\qquad i = 1,...,s, \; j = 1,...,m \quad .
\end{aligned}$$

We now state qualification conditions guaranteeing the convergence of (MSM), similar to those needed for the Han-Lou method.

Proposition 5.1. Suppose (HLP) is solvable, and fix any $\lambda > 0$. Let σ be a permutation on the set $\{1,...,s\}$ such that for some $q \in \{1,...,s\}$, $C_{\sigma(q+1)}, C_{\sigma(q+2)}, \ldots, C_{\sigma(s)}$ are polyhedral. If

$$\text{ri}(\text{dom } h) \cap \left(\bigcap_{i=0}^{q} \text{ri}(C_{\sigma(i)}) \right) \cap \left(\bigcap_{i=q+1}^{s} C_{\sigma(i)} \right) \neq \varnothing \quad ,$$

or h is pseudopolyhedral and

$$\text{dom } h \cap \left(\bigcap_{i=0}^{q} \text{ri}(C_{\sigma(i)}) \right) \cap \left(\bigcap_{i=q+1}^{s} C_{\sigma(i)} \right) \neq \varnothing \quad ,$$

then the sequence $\{\mathbf{x}^k\}$ generated by the method (MSM) converges to a solution of (HLP).

Proof. By Rockafellar (1970a), Theorem 23.8, the stated conditions on h and $C_0,...,C_s$ are sufficient to imply

$$\partial\left[h + \sum_{i=1}^{s} \delta_{C_i}\right] = \partial h + \sum_{i=1}^{s} \partial\delta_{C_i} = \partial h + \sum_{i=1}^{s} N_{C_i} \quad .$$

Suppose that $\mathbf{x}^*$ is a solution to (HLP). Then

$$\partial h(\mathbf{x}^*) + \sum_{i=1}^{s} N_{C_i}(\mathbf{x}^*) \ni \mathbf{0}$$

and there must exist $\mathbf{p}_1^* \in N_{C_1}(\mathbf{x}^*), \ldots, \mathbf{p}_s^* \in N_{C_s}(\mathbf{x}^*)$ such that $-\sum_{i=1}^{s} \mathbf{p}_i^* \in \partial h(\mathbf{x}^*)$. Let $\mathbf{p}^* \in (\mathbb{R}^n)^s$ denote the vector $(\mathbf{p}_1^*,...,\mathbf{p}_s^*)$. Then $-\sum_{i=1}^{s} \mathbf{p}_i^* = -\mathbf{M}^T\mathbf{p}^* \in \partial f(\mathbf{x}^*)$ and

$$\mathbf{p}^* = (\mathbf{p}_1^*,...,\mathbf{p}_s^*) \in N_{C_1}(\mathbf{x}^*) \times ... \times N_{C_s}(\mathbf{x}^*) = \partial g(\mathbf{M}\mathbf{x}^*) \ ,$$

so

$$(\mathbf{x}^*, -\mathbf{M}^T\mathbf{p}^*) \in \partial f \qquad (\mathbf{M}\mathbf{x}^*, \mathbf{p}^*) \in \partial g \ .$$

By Proposition 3.42, the existence of such a Kuhn-Tucker pair $(\mathbf{x}^*, \mathbf{p}^*)$ implies that the alternating direction method of multipliers is convergent. As (MSM) is an implementation of the alternating direction method of multipliers for this particular choice of f, g, and $\mathbf{M}$, it too is convergent. The sequence $\{\mathbf{x}^k\}$ converges to a solution of (P), which is identical to a solution of (HLP). ■

The convergence properties of the method (MSM) are more general than those of the Han-Lou algorithm, which also requires that h be strongly convex and that the stepsize λ be sufficiently small. These differences are directly attributable to (MSM) being based on

Douglas-Rachford splitting (by way of the alternating direction method of multipliers), while the Han-Lou method is an instance of forward-backward splitting, which is less robust. In fact, the Han-Lou method can be implemented in exactly the same way as (MSM), except for the calculation of $\mathbf{x}^{k+1}$, which in the Han-Lou case is

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \left\{ h(\mathbf{x}) + \langle \sum_{i=1}^{s} \mathbf{p}_i^k, \mathbf{x} \rangle \right\} = \nabla h^* \left(-\sum_{i=1}^{s} \mathbf{p}_i^k \right) \quad .$$

The only advantage of the Han-Lou approach would be in cases where h is differentiable and strongly convex, and the calculation $\mathbf{x}^{k+1} = \nabla h^*(-\sum_{i=1}^{s} \mathbf{p}_i)$ can be done easily, perhaps in closed form, but the corresponding augmented Lagrangian calculation used to calculate $\mathbf{x}^{k+1}$ in (MSM) is for some reason significantly more difficult.

(MSM) has some potential for parallelism, as the computation of $\mathbf{z}_i^{k+1}$ may be done in parallel over i, and the calculation of p_{ij}^{k+1} in parallel over both i and j. The computation of $\mathbf{x}^{k+1}$ cannot in general be parallelized, and thus presents a potential "Amdahl's law" bottleneck (*e.g.* Bertsekas and Tsitsiklis 1989, Chapter 1). However, if h has separable structure, that is

$$h(\mathbf{x}) = \sum_{j=1}^{n} h_j(x_j) \quad ,$$

where $h_j \colon \mathbb{R} \to (-\infty, +\infty]$ are closed proper convex, then the $\mathbf{x}$ minimization decomposes into

$$x_j^{k+1} = \arg\min_{x} \left\{ h_j(x) + \left(\sum_{i=1}^{s} p_{ij}^k \right) x + \frac{s\lambda}{2} \left(x - \frac{1}{s} \sum_{i=1}^{s} z_{ij}^k \right)^2 \right\} \qquad j = 1,\ldots,n \quad .$$

These calculations may be performed in parallel over j. A similar decomposition occurs in cases where h has *block-separable* structure, as defined in the next section.

Finally, we note that (MSM) may be applied to general convex programs of the form

$$\begin{array}{ll} \min & h_0(\mathbf{x}) \\ \text{ST} & h_i(\mathbf{x}) \leq 0, \quad i=1,...,s \end{array} ,$$

where $h_0,...,h_s$: $\mathbb{R}^n \rightarrow (-\infty, +\infty]$ are closed proper convex, by setting

$$C_i = \{\mathbf{x} \in \mathbb{R}^n \mid h_i(\mathbf{x}) \leq 0\}, \quad i=1,...,s \quad .$$

However, unless the h_i have special form, performing projection onto the C_i may be difficult. In the next two sections, we will examine alternative decomposition methods that circumvent this difficulty in the case that $h_0,...,h_s$ have block-separable structure. These methods involve constructing a somewhat more sophisticated splitting than the one used to generate (MSM).

5.2. Spingarn's Method for Block-Separable Problems

We will now present Spingarn's (1985b) decomposition algorithm for block-separable convex programming. The development will differ somewhat from Spingarn's, so that it can be combined with the theory of the epigraphic projection method. First, we define the notion of block separability.

Let $\{1,...,n\}$ be partitioned into $d \geq 2$ nonempty subsets N_j, $j=1,...,d$, and let $n_j = |N_j|$ for all j. For any $\mathbf{x} \in \mathbb{R}^n$, let $\mathbf{x}_j \in \mathbb{R}^{n_j}$ denote the subvector of $\mathbf{x}$ with components x_q, $q \in N_j$.

Definition 5.1. A convex program is called *block separable* if for some such partition, it takes the form

$$\begin{aligned} \text{minimize} \quad & \sum_{j=0}^{d} h_{0j}(\mathbf{x}_j) \\ \text{subject to} \quad & \sum_{j=0}^{d} h_{ij}(\mathbf{x}_j) \leq 0, \; i = 1,\ldots,m \quad , \end{aligned} \tag{BSCP}$$

where h_{ij} are convex functions on $\mathbb{R}^{n_j}$ for i=0,...,m and j=1,...,d.

In the case $d = n$ and $N_j = \{j\}$ for all j, a block-separable convex program is separable. Thus, the class of all block separable convex programs includes the class of separable convex programs. To simplify the discussion, we require that the convex functions h_{ij}, i=1,...,m and j=1,...,d be everywhere finite-valued. For the h_{0j}, we loosen this requirement somewhat and allow each h_{0j} to be of the form $\bar{h}_{0j} + \delta_{C_j}$, where the $C_j \subseteq \mathbb{R}^{n_j}$, j=1,...,d, are closed, nonempty, and convex, and the $\bar{h}_{0j}$ are finite-valued and convex throughout $\mathbb{R}^{n_j}$. Let $C = C_1 \times \ldots \times C_d = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x}_j \in C_j, \; j=1,\ldots,d\} \subseteq \mathbb{R}^n$.

To obtain a useful decomposition of (BSCP), we do not propose to put it immediately in the form (P), but to express it as a minimum problem for a sum of *three* convex functions. First, we introduce some notation: let the components of vectors $\mathbf{u} \in \mathbb{R}^{md} \approx (\mathbb{R}^m)^d$ be written u_{ij}, for i=1,...,m and j=1,...,d. Similarly, denote the components of vectors $\mathbf{z} \in \mathbb{R}^{mn} \approx (\mathbb{R}^m)^n$ by z_{iq}, for i=1,...,m and q=1,...,n, and let $\mathbf{H}$ be the $mn \times n$ matrix taking vectors $\mathbf{x} \in \mathbb{R}^n$ to $\mathbf{z} \in \mathbb{R}^{mn}$, where $z_{iq} = x_q$ for all i and q. That is,

$$\mathbf{H} = \underbrace{\left[\begin{array}{c} \mathbf{I} \\ \mathbf{I} \\ \vdots \\ \mathbf{I} \end{array}\right]}_{m \text{ times}} \quad .$$

n columns

For any $\mathbf{z} \in \mathbb{R}^{mn}$, let $\mathbf{z}_{ij}$ denote the subvector of $\mathbf{z}$ consisting of the z_{iq}, $q \in N_j$. Now define $F_1: \mathbb{R}^n \to (-\infty, +\infty]$, $F_2: \mathbb{R}^{md} \to (-\infty, +\infty]$, and $F_3: \mathbb{R}^{mn} \times \mathbb{R}^{md} \to (-\infty, +\infty]$ as follows:

$$F_1(\mathbf{x}) = \sum_{j=1}^{d} h_{0j}(\mathbf{x}_j)$$

$$F_2(\mathbf{u}) = \begin{cases} 0, & \sum_{j=0}^{d} u_{ij} = 0 \ \ \forall\ 1 \le i \le m \\ +\infty, & \text{otherwise} \end{cases}$$

$$F_3(\mathbf{z}, \mathbf{u}) = \begin{cases} 0, & h_{ij}(\mathbf{z}_{ij}) \le u_{ij} \ \ \forall\ 1 \le i \le m, 1 \le j \le d \\ +\infty, & \text{otherwise} \end{cases} .$$

F_2 and F_3 are both indicator functions of nonempty, closed convex sets, and, in particular, F_2 is the indicator functions for a linear subspace of $\mathbb{R}^{md}$. F_1 is the objective function of the original problem. Therefore, all three functions are closed proper convex.

Proposition 5.2. A vector $\mathbf{x} \in \mathbb{R}^n$ is feasible for (BSCP) if and only if there exists $\mathbf{u} \in \mathbb{R}^{md}$ such that $(\mathbf{x}, \mathbf{u})$ is feasible for

$$\begin{array}{ll} \text{minimize} & F_1(\mathbf{x}) + F_2(\mathbf{u}) + F_3(\mathbf{H}\mathbf{x}, \mathbf{u}) \\ \text{subject to} & \mathbf{x} \in \mathbb{R}^n \\ & \mathbf{u} \in \mathbb{R}^{md} \quad , \end{array} \tag{P3}$$

that is, $F_1(\mathbf{x}) + F_2(\mathbf{u}) + F_3(\mathbf{H}\mathbf{x}, \mathbf{u})$ is finite-valued. Furthermore, $\mathbf{x} \in \mathbb{R}^n$ is optimal for (BSCP) if and only if there exists $\mathbf{u} \in \mathbb{R}^{md}$ such that $(\mathbf{x}, \mathbf{u})$ is optimal for (P3).

Proof. Assume $\mathbf{x}$ is feasible for (BSCP), that is, $h_{0j}(\mathbf{x}_j) < \infty$ for all j, and $\sum_{j=1}^{d} h_{ij}(\mathbf{x}_j) \le 0$ for all i. Let

$$u_{ij} \ = \ h_{ij}(\mathbf{x}_j) - \frac{1}{d}\sum_{l=1}^{d} h_{il}(\mathbf{x}_l) \ \ge \ h_{ij}(\mathbf{x}_j)$$

for all $i = 1,...,m$ and $j=1,...,d$. Then

$$\sum_{j=1}^{d} u_{ij} = \sum_{j=1}^{d} h_{ij}(\mathbf{x}_l) - \sum_{l=1}^{d} h_{il}(\mathbf{x}_l) = 0$$

for all i, and so $F_2(\mathbf{u}) = 0$. Also, $h_{ij}(\mathbf{x}_j) \leq u_{ij}$ for all i and j, so $F_3(\mathbf{Hx}, \mathbf{u}) = 0$. As $\sum_{j=1}^{d} h_{0j}(\mathbf{x}_j) = F_1(\mathbf{x}) < \infty$ by the feasibility of $\mathbf{x}$ for (BSCP), we have

$$F_1(\mathbf{x}) + F_2(\mathbf{u}) + F_3(\mathbf{Hx}, \mathbf{u}) < \infty \quad .$$

Conversely, let $(\mathbf{x}, \mathbf{u})$ be feasible for (P3). Then $\sum_{j=1}^{d} h_{0j}(\mathbf{x}_j) = F_1(\mathbf{x}) < \infty$, so $h_{0j}(\mathbf{x}_j) < \infty$ for all j. Also, $F_3(\mathbf{Hx}, \mathbf{u}) < \infty$ implies $h_{ij}(\mathbf{x}_j) \leq u_{ij}$ for all i and j, and $F_2(\mathbf{u}) < \infty$ implies that $\sum_{j=1}^{d} u_{ij} = 0$ for all i. Therefore, $\sum_{j=1}^{d} h_{ij}(\mathbf{x}_j) \leq \sum_{j=1}^{d} u_{ij} = 0$ for all i, and $\mathbf{x}$ is feasible for (BSCP).

Remarking that F_2 and F_3 can only take the values 0 and $+\infty$, we then have

$$\sum_{j=1}^{d} h_{0j}(\mathbf{x}_j) = \min_{\mathbf{u} \in \mathbb{R}^{md}} \{F_1(\mathbf{x}) + F_2(\mathbf{u}) + F_3(\mathbf{Hx}, \mathbf{u})\} \quad ,$$

for all $\mathbf{x}$ feasible for (BSCP), thereby proving the assertion about optimality. ■

A problem such as (P3) would seem a natural candidate for an operator splitting method that would find the zero of a sum of *three* operators. Satisfactory schemes of this type are presently unknown, and a topic for future research (the n-fold backward scheme of Passty (1979), having only ergodic convergence, cannot be considered very practical). To apply a two-operator splitting technique, one natural approach is to put (P3) into the form (P) by summing two of the functions F_1, F_2, and F_3 into one.

We now show that Spingarn's (1985b) decomposition approach for (BSCP) may be obtained by converting (P3) to the form (P) by setting $f(\mathbf{x}, \mathbf{u}) = F_1(\mathbf{x}) + F_3(\mathbf{Hx}, \mathbf{u})$, $g(\mathbf{x}, \mathbf{u}) = F_2(\mathbf{u})$, and letting $\mathbf{M}$ be the $(n+md)\times(n+md)$ identity matrix. An alternate expression for $f(\mathbf{x}, \mathbf{u})$ is then

$$f(\mathbf{x}, \mathbf{u}) = \begin{cases} \sum_{j=1}^{d} h_{0j}(\mathbf{x}_j), & h_{ij}(\mathbf{x}_j) \le u_{ij} \ \ \forall\ 1 \le i \le m,\ 1 \le j \le d \\ +\infty, & \text{otherwise} \quad . \end{cases}$$

Note also that $g(\mathbf{x}, \mathbf{u}) = \delta_V(\mathbf{x}, \mathbf{u})$, where V is the linear subspace

$$V = \{(\mathbf{x}, \mathbf{u}) \in \mathbb{R}^n \times \mathbb{R}^{md} \mid \sum_{j=1}^{d} u_{ij} = 0 \ \ \forall\ 1 \le i \le m\} \quad .$$

Because the orthogonal complement of the null space $\mathcal{N}(\mathbf{A})$ of a matrix $\mathbf{A}$ is $\mathrm{im}(\mathbf{A}^T)$ (see *e.g.* Strang 1976), we can derive that the orthogonal complement of V is

$$V^{\perp} = \{(\mathbf{0}, \mathbf{p}) \in \mathbb{R}^n \times \mathbb{R}^{md} \mid p_{i1} = p_{i2} = \ldots = p_{id},\ i = 1,\ldots,m\} \quad .$$

We now apply the alternating direction method of multipliers (ADMOM$_\lambda$) to

$$f(\mathbf{x}, \mathbf{u}) = F_1(\mathbf{x}) + F_3(\mathbf{Hx}, \mathbf{u}) \qquad \text{and} \qquad g(\mathbf{x}, \mathbf{u}) = \delta_V(\mathbf{x}, \mathbf{u})$$

with $\mathbf{M} = \mathbf{I}$. As $\mathbf{M} = \mathbf{I}$, we know from Section 3.5.6 that we could equally well apply the primal Douglas-Rachford method, obtaining an algorithm that would be identical, except that the roles of λ and $\frac{1}{\lambda}$ would be reversed.

The role of the "$\mathbf{x}^k$" variables of (ADMOM$_\lambda$) will be played by pairs $(\mathbf{x}^k, \mathbf{u}^k) \in \mathbb{R}^n \times \mathbb{R}^{md}$, the place of the "$\mathbf{z}^k$" variables will be taken by pairs $(\mathbf{y}^k, \mathbf{v}^k) \in \mathbb{R}^n \times \mathbb{R}^{md}$, and the role of

the multipliers "$\mathbf{p}^k$" will be assumed by pairs $(\mathbf{q}^k, \mathbf{p}^k) \in \mathbb{R}^n \times \mathbb{R}^{md}$. Noting from the derivation of the alternating direction method of multipliers (see Section 3.5.6) that

$$((\mathbf{q}^k, \mathbf{p}^k), (\mathbf{y}^k, \mathbf{v}^k)) \in B = \partial g^* = (V \times V^\perp)^{-1} = V^\perp \times V \quad ,$$

we have, for all $k \geq 0$, that $\mathbf{q}^k = \mathbf{0}$ and there exists some $\pi^k \in \mathbb{R}^m$ such that

$$p_{i1}{}^k = p_{i2}{}^k = \ldots = p_{id}{}^k = \pi_i{}^k \quad , \qquad i = 1,\ldots,m. \tag{*}$$

Similarly, $\sum_{j=1}^{d} v_{ij} = 0$ for all i. To begin the alternating direction method of multipliers iteration, we suppose we are given some such vector of multipliers $\mathbf{p}^k \in \mathbb{R}^{md}$ satisfying (*) along with some π^k, and some $(\mathbf{y}^k, \mathbf{v}^k) \in \mathbb{R}^n \times \mathbb{R}^{md}$ such that $\sum_{j=1}^{d} v_{ij} = 0$ for all i. Using that $\mathbf{M} = \mathbf{I}$ and completing the square, the ensuing minimization involving the function f can be written

$$\begin{aligned}
(\mathbf{x}^{k+1}, \mathbf{u}^{k+1}) &= \underset{(\mathbf{x}, \mathbf{u})}{\arg\min} \{f(\mathbf{x}, \mathbf{u}) + \langle (\mathbf{0}, \mathbf{p}^k), (\mathbf{x}, \mathbf{u}) \rangle + \tfrac{\lambda}{2} \| (\mathbf{x}, \mathbf{u}) - (\mathbf{y}^k, \mathbf{v}^k) \|^2 \} \\
&= \underset{(\mathbf{x}, \mathbf{u})}{\arg\min} \{f(\mathbf{x}, \mathbf{u}) + \tfrac{\lambda}{2} \| (\mathbf{x}, \mathbf{u}) - (\mathbf{y}^k, \mathbf{v}^k - \tfrac{1}{\lambda}\mathbf{p}^k) \|^2 \} \\
&= \underset{(\mathbf{x}, \mathbf{u})}{\arg\min} \{f(\mathbf{x}, \mathbf{u}) + \tfrac{\lambda}{2} \| \mathbf{x} - \mathbf{y}^k \|^2 + \tfrac{\lambda}{2} \| \mathbf{u} - (\mathbf{v}^k - \tfrac{1}{\lambda}\mathbf{p}^k) \|^2 \} \quad .
\end{aligned}$$

Now, the minimand above can be rewritten as $\sum_{j=1}^{d} \phi_j(\mathbf{x}_j, u_{1j}, \ldots, u_{mj})$, where

$$\phi_j(\mathbf{x}_j, u_{1j}, \ldots, u_{mj}) = \begin{cases} h_{0j}(\mathbf{x}_j) + \frac{\lambda}{2}\|\mathbf{x}_j - \mathbf{y}_j^k\|^2 + \frac{\lambda}{2} \sum_{i=1}^{m} (u_{ij} - v_{ij}^k + \frac{1}{\lambda}\pi_i^k)^2, & h_{ij}(\mathbf{x}_j) \leq u_{ij}, \ \forall\ i \\ +\infty \ , & \text{otherwise} \end{cases} \quad .$$

Given $\mathbf{x}_j$, each ϕ_j achieves its minimum over u_{ij} at $u_{ij} = v_{ij}{}^k - \frac{1}{\lambda}\pi_i{}^k$ if $h_{ij}(\mathbf{x}_j) \leq v_{ij}{}^k - \frac{1}{\lambda}\pi_i$, and at $u_{ij} = h_{ij}(\mathbf{x}_j)$ otherwise. Therefore, the minimization over $(\mathbf{x}, \mathbf{u})$ may be written

$$\mathbf{x}_j^{k+1} = \arg\min_{\mathbf{x}_j \in \mathbb{R}^{N_j}} \left\{ h_{0j}(\mathbf{x}_j) + \frac{\lambda}{2}\|\mathbf{x}_j - \mathbf{y}_j^k\| + \frac{\lambda}{2}\sum_{i=1}^{m} \max^2\left\{0,\ h_{ij}(\mathbf{x}_j) - v_{ij}^k + \frac{1}{\lambda}\pi_i^k\right\}\right\} \quad j=1,\ldots,n$$

$$u_{ij}^{k+1} = \max\left\{v_{ij}^k + \frac{1}{\lambda}\pi_i^k,\ h_{ij}(\mathbf{x}_j^{k+1})\right\} \qquad \begin{matrix} i=1,\ldots,m, \\ j=1,\ldots,n\ . \end{matrix}$$

Note that the first step has been decomposed into d independent calculations of $\mathbf{x}_j^{k+1}$, $j=1,\ldots,d$, which can be carried out in parallel, followed by md potentially parallel calculations of the u_{ij}^{k+1}.

Again completing the square and using that $\mathbf{M} = \mathbf{I}$, the second step of the alternating direction method of multipliers, the minimization involving g, may now be written

$$\begin{aligned}(\mathbf{y}^{k+1}, \mathbf{v}^{k+1}) &= \arg\min_{(\mathbf{y},\mathbf{v})} \{g(\mathbf{y},\mathbf{v}) - \langle(\mathbf{0}, \mathbf{p}^k), (\mathbf{y},\mathbf{v})\rangle + \tfrac{\lambda}{2}\|(\mathbf{y},\mathbf{v}) - (\mathbf{x}^{k+1}, \mathbf{u}^{k+1})\|^2\} \\ &= \arg\min_{(\mathbf{y},\mathbf{v})} \{g(\mathbf{y},\mathbf{v}) + \tfrac{\lambda}{2}\|(\mathbf{y},\mathbf{v}) - (\mathbf{x}^{k+1}, \mathbf{u}^{k+1} + \tfrac{1}{\lambda}\mathbf{p}^k)\|^2\} \\ &= \arg\min_{(\mathbf{y},\mathbf{v}) \in V} \{\tfrac{\lambda}{2}\|\mathbf{y} - \mathbf{x}^{k+1}\|^2 + \tfrac{\lambda}{2}\|\mathbf{v} - (\mathbf{u}^{k+1} + \tfrac{1}{\lambda}\mathbf{p}^k)\|^2\}\ ,\end{aligned}$$

which amounts to projecting the point $(\mathbf{x}^{k+1}, \mathbf{u}^{k+1} + \frac{1}{\lambda}\mathbf{p}^k)$ onto V to obtain $(\mathbf{y}^{k+1}, \mathbf{v}^{k+1})$. Because of the simple form of V, this operation can be performed by setting $\mathbf{y}^{k+1} = \mathbf{x}^{k+1}$ and

$$v_{ij}^{k+1} = u_{ij}^{k+1} + \pi_i^k - \frac{1}{d}\sum_{l=1}^{d}(u_{il}^{k+1} + \pi_i^k) = u_{ij}^{k+1} - \frac{1}{d}\sum_{l=1}^{d} u_{il}^{k+1}$$

for all i and j. The multiplier update formula now conveniently reduces from

$$(\mathbf{0}, \mathbf{p}^{k+1}) = (\mathbf{0}, \mathbf{p}^k) + \lambda((\mathbf{x}^{k+1}, \mathbf{u}^{k+1}) - (\mathbf{y}^{k+1}, \mathbf{v}^{k+1}))$$

to

$$\pi_i^{k+1} = \pi_i^k + \lambda\,\frac{1}{d}\sum_{l=1}^{d} u_{il}^{k+1}\ .$$

Eliminating the variables $\{\mathbf{y}^k\}$, which evolve identically to the $\{\mathbf{x}^k\}$, the entire alternating direction method of multipliers now becomes *Spingarn's block-separable method,*

$$\mathbf{x}_j^{k+1} = \arg\min_{\mathbf{x}_j \in \mathbb{R}^{N_j}} \left\{ h_{0j}(\mathbf{x}_j) + \frac{\lambda}{2}\|\mathbf{x}_j - \mathbf{x}_j^k\|^2 + \frac{\lambda}{2}\sum_{i=1}^{m} \max^2\left\{0,\ h_{ij}(\mathbf{x}_j) - v_{ij}^k + \frac{1}{\lambda}\pi_i^k\right\}\right\} \qquad j=1,...,d$$

$$u_{ij}^{k+1} = \max\left\{v_{ij}^k + \frac{1}{\lambda}\pi_i^k,\ h_{ij}(\mathbf{x}_j^{k+1})\right\} \qquad \begin{matrix} i=1,...m \\ j=1,...d \end{matrix}$$

$$v_{ij}^{k+1} = u_{ij}^{k+1} - \frac{1}{d}\sum_{l=1}^{d} u_{il}^{k+1} \qquad \begin{matrix} i=1,...m \\ j=1,...d \end{matrix}$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda}{d}\sum_{l=1}^{d} u_{il}^{k+1} \qquad \begin{matrix} i=1,...m \\ j=1,...d \end{matrix} .$$

The sequences $\{\mathbf{x}^k\}$, $\{\mathbf{v}^k\}$, $\{\mathbf{u}^k\}$, $\{\pi^k\}$ are in $\mathbb{R}^n$, $\mathbb{R}^{md}$, $\mathbb{R}^{md}$, and $\mathbb{R}^m$, respectively. Arbitrary initial values may be supplied for $\mathbf{x}^0$, $\mathbf{v}^0$, and π^0. The variables $u_{ij}{}^k$ are temporary, intermediate results, and do not have to carried over from one iteration to the next.

Spingarn (1985b) derived an identical algorithm from the method of partial inverses. This derivation is fundamentally the same to the one given above, because $g(\mathbf{x}, \mathbf{u})$ is the indicator function of a linear subspace V. We showed in Chapter 3 that the alternating direction method of multipliers is a special case of Douglas-Rachford splitting, as applied to the operators $A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$. Here, ∂g^* is the normal cone operator of the linear subspace $V^\perp$. The method of partial inverses is equivalent to the special case of Douglas-Rachford splitting in which one of the operators is the normal cone of a linear subspace, so the two derivations are essentially identical.

We will not duplicate Spingarn's convergence proof for his block-separable method, but simply state his final result in simplified form:

Proposition 5.3. Consider the problem (BSCP), under the assumption that the convex functions h_{ij} are everywhere finite-valued convex for $i = 1,...,m$, and that the h_{0j} are of the form $\bar{h}_{0j} + \delta_{C_j}$, where the $C_j \subseteq \mathbb{R}^{n_j}$, $j=1,...,d$, are closed, nonempty, and convex, and the $\bar{h}_{0j}$ are finite-valued and convex throughout $\mathbb{R}^{n_j}$. Suppose that there exists some point $\tilde{\mathbf{x}}$ satisfying the *Slater condition* $\tilde{\mathbf{x}} \in \text{ri}(C) = \text{ri}(C_1) \times ... \times \text{ri}(C_d)$ and $\sum_{j=1}^{d} h_{ij}(\tilde{\mathbf{x}}_j) < 0$ for $i = 1,...,m$. Then if (BSCP) is solvable, the sequence $\{\mathbf{x}^k\}$ generated by Spingarn's block separable method converges to a solution of (BSCP).

Fundamentally, the Slater condition is to insure that $\partial[f + g] = \partial f + \partial g$, so that a splitting method can successfully attack the problem. The empirical properties of the block-separable method appear to be unknown.

As a final note, the method could be generalized by using the method of Proposition 4.7 instead of (ADMOMλ) in the derivation. Such an approach would permit varying over- and under-relaxation, as well as approximate implementation of the minimization step.

5.3. The Epigraphic Projection Method

It may be argued that Spingarn's block-separable method does not take full advantage of the structure of problem (BSCP). In particular, even when all the functions h_{ij} have a simple form, the individual functions

$$\chi_j(\mathbf{x}_j\,;\,\mathbf{x}_j^k,\mathbf{u},\,\mathbf{p}) = h_{0j}(\mathbf{x}_j) + \frac{\lambda}{2}\|\mathbf{x}_j - \mathbf{x}_j^k\|^2 + \frac{\lambda}{2}\sum_{i=1}^{m} \max^2\left\{0,\, h_{ij}(\mathbf{x}_j) - v_{ij}^k + \frac{1}{\lambda}\pi_i^k\right\}\,,$$

which must be minimized in $\mathbf{x}_j$ at each iteration, can have a complicated structure. Even in the simplest case, where, $n_j = 1$, $\chi_j(\,\cdot\,;\,\mathbf{x}_j^k,\mathbf{u},\,\mathbf{p})$ may have as many as m "breakpoints", and for larger n_j, the situation may be much more complex. Thus, for large m, the calculation of

the $\mathbf{x}_j^{k+1}$ may be cumbersome. We will now propose a new method in which no minimization involves more than one of the functions h_{ij} at a time. The basic idea is to combine F_1 and F_2 rather than F_1 and F_3.

We argue that combining F_1 and F_2 is a much more natural way to apply a two-operator splitting scheme to problems of the form (P3) than combining F_1 and F_3. The reason is that F_1 depends only on $\mathbf{x}$ and F_2 depends only on $\mathbf{u}$, so the proximal step for F_1+F_2 decomposes into two independent calculations, one for $\mathbf{x}$ and one for $\mathbf{u}$, thus preserving the three-way splitting implicit in the formulation (P3). Spingarn's approach does not preserve this independence because it combines F_1 with F_3, which depends on *both* $\mathbf{x}$ and $\mathbf{u}$. The reason that Spingarn pursued such an approach was that the splitting method that he had available, the method of partial inverses, can in essence only handle situations in which one operator is the normal cone map of a linear subspace. In terms of the formulation of the previous section, then, Spingarn could not combine F_2, which corresponds to a linear subspace cone operator, with either F_1 or F_3, which do not.

We now set up our alternative method as an application of the fixed-stepsize alternating direction method of multipliers (ADMOMλ). First, let

$$f(\mathbf{x}, \mathbf{u}) = F_1(\mathbf{x}) + F_2(\mathbf{u}) = \begin{cases} \sum_{j=1}^{d} h_{0j}(\mathbf{x}_j), & \sum_{j=1}^{d} u_{ij} = 0 \ \ \forall\ 1 \le i \le m \\ +\infty, & \text{otherwise} \quad . \end{cases}$$

Then let $\mathbf{M}$ be the $(n+md) \times (mn+md)$ matrix taking $(\mathbf{x}, \mathbf{u}) \in \mathbb{R}^n \times \mathbb{R}^{md}$ to $(\mathbf{z}, \mathbf{u}) \in \mathbb{R}^{mn} \times \mathbb{R}^{md}$, where $z_{iq} = x_q$ for all i and q, that is,

$$\mathbf{M} = \begin{bmatrix} \mathbf{H} \\ \mathbf{I} \end{bmatrix} \begin{matrix} \} \, nd \text{ rows} \\ \} \, md \text{ rows} \end{matrix} ,$$

where $\mathbf{H}$ is as defined in the previous section. Because $\mathbf{H}$ has full column rank, so does $\mathbf{M}$. Finally let $g: \mathbb{R}^{mn} \times \mathbb{R}^{md} \to (-\infty, +\infty]$ be given by

$$g(\mathbf{z}, \mathbf{u}) = F_3(\mathbf{z}, \mathbf{u}) = \begin{cases} 0, & h_{ij}(\mathbf{z}_{ij}) \le u_{ij} \quad \forall \ 1 \le i \le m, \ 1 \le j \le d \\ +\infty, & \text{otherwise} \end{cases} .$$

We then have a problem in the form

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}, \mathbf{u}) + g(\mathbf{M}(\mathbf{x}, \mathbf{u})) \\ &\text{subject to} \quad (\mathbf{x}, \mathbf{u}) \in \mathbb{R}^n \times \mathbb{R}^{md} \quad , \end{aligned} \tag{PXU}$$

equivalent to (P3), and hence to (BSCP). We now apply the alternating direction method of multipliers. The role of the "$\mathbf{x}^k$" variables will be played by pairs $(\mathbf{x}^k, \mathbf{u}^k) \in \mathbb{R}^n \times \mathbb{R}^{md}$, while the role of the "$\mathbf{z}^k$" variables will now be played by pairs $(\mathbf{z}^k, \mathbf{v}^k) \in \mathbb{R}^{mn} \times \mathbb{R}^{md}$. The multiplier vectors now take the form $(\mathbf{p}^k, \mathbf{q}^k) \in \mathbb{R}^{mn} \times \mathbb{R}^{md}$.

Consider first the "f" minimization required by (ADMOM$_\lambda$), namely

$$(\mathbf{x}^{k+1}, \mathbf{u}^{k+1}) = \underset{(\mathbf{x}, \mathbf{u})}{\arg\min} \left\{ f(\mathbf{x}, \mathbf{u}) + \langle (\mathbf{p}^k, \mathbf{q}^k), \mathbf{M}(\mathbf{x}, \mathbf{u}) \rangle + \frac{\lambda}{2} \| \mathbf{M}(\mathbf{x}, \mathbf{u}) - (\mathbf{z}^k, \mathbf{v}^k) \|^2 \right\} .$$

Because of the special structure of $\mathbf{M}$ and the definition $f(\mathbf{x}, \mathbf{u}) = F_1(\mathbf{x}) + F_2(\mathbf{u})$, this calculation decomposes into

$$\begin{aligned} \mathbf{x}^{k+1} &= \underset{\mathbf{x}}{\arg\min} \left\{ F_1(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{H}\mathbf{x} \rangle + \frac{\lambda}{2} \| \mathbf{H}\mathbf{x} - \mathbf{z}^k \|^2 \right\} \\ \mathbf{u}^{k+1} &= \underset{\mathbf{u}}{\arg\min} \left\{ F_2(\mathbf{u}) + \langle \mathbf{q}^k, \mathbf{u} \rangle + \frac{\lambda}{2} \| \mathbf{u} - \mathbf{v}^k \|^2 \right\} . \end{aligned}$$

The special structure of F_1 and $\mathbf{H}$ yields that an equivalent calculation of $\mathbf{x}^{k+1}$ is

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \{ \sum_{j=1}^{d} h_{0j}(\mathbf{x}_j) + \sum_{j=1}^{d} \langle \sum_{i=1}^{m} \mathbf{p}_{ij}^k, \mathbf{x}_j \rangle + \frac{\lambda}{2} \sum_{j=1}^{d} \sum_{i=1}^{m} \| \mathbf{x}_j - \mathbf{z}_{ij}^k \|^2 \} \ .$$

Decomposing over j and rewriting the quadratic terms, one obtains

$$\mathbf{x}_j^{k+1} = \arg\min_{\mathbf{x}_j \in \mathbb{R}^{N_j}} \{ h_{0j}(\mathbf{x}_j) + \langle \sum_{i=1}^{m} \mathbf{p}_{ij}^k, \mathbf{x}_j \rangle + \frac{m\lambda}{2} \| \mathbf{x}_j - \frac{1}{m} \sum_{i=1}^{m} \mathbf{z}_{ij}^k \|^2 \} \ , \quad j=1,...,d.$$

The calculation of $\mathbf{u}^{k+1}$ may be rewritten

$$\mathbf{u}^{k+1} = \arg\min_{\mathbf{u}} \{ F_2(\mathbf{u}) + \frac{\lambda}{2} \| \mathbf{u} - (\mathbf{v}^k - \frac{1}{\lambda} \mathbf{q}^k) \|^2 \} \ ,$$

that is, projection of $\mathbf{v}^k - \frac{1}{\lambda}\mathbf{q}^k$ onto the linear space $\{ \mathbf{u} \in \mathbb{R}^{md} \mid \sum_{j=1}^{d} u_{ij} = 0, \ i = 1,...,m \}$. This task decomposes into m independent projections of $v_{i1}^k - \frac{1}{\lambda} q_{i1}^k, ..., v_{id}^k - \frac{1}{\lambda} q_{id}^k$ onto the spaces $\{ \mathbf{u}_i \in \mathbb{R}^d \mid u_{i1} + ... + u_{id} = 0 \}$. These operations may be implemented via

$$u_{ij}^{k+1} = v_{ij}^k - \frac{1}{\lambda} q_{ij}^k - \sum_{l=1}^{d} (v_{il}^k - \frac{1}{\lambda} q_{il}^k) = \left(v_{ij}^k - \sum_{l=1}^{d} v_{il}^k \right) - \frac{1}{\lambda} \left(q_{ij}^k - \sum_{l=1}^{d} q_{il}^k \right)$$

for $i = 1,...,m$ and $j=1,...,d$.

We now consider the "g" minimization of (ADMOM$_\lambda$), which in this case becomes

$$\begin{aligned} (\mathbf{z}^{k+1}, \mathbf{v}^{k+1}) &= \arg\min_{(\mathbf{z}, \mathbf{v})} \{ g(\mathbf{z}, \mathbf{v}) - \langle (\mathbf{p}^k, \mathbf{q}^k), (\mathbf{z}, \mathbf{v}) \rangle + \frac{\lambda}{2} \| (\mathbf{z}, \mathbf{v}) - \mathbf{M}(\mathbf{x}^{k+1}, \mathbf{u}^{k+1}) \|^2 \} \\ &= \arg\min_{(\mathbf{y}, \mathbf{v})} \{ g(\mathbf{z}, \mathbf{v}) + \frac{\lambda}{2} \| (\mathbf{z}, \mathbf{v}) - [\mathbf{M}(\mathbf{x}^{k+1}, \mathbf{u}^{k+1}) + \frac{1}{\lambda} (\mathbf{p}^k, \mathbf{q}^k)] \|^2 \} \ , \end{aligned}$$

For all i and j, let $g_{ij}: \mathbb{R}^{n_j} \to [0, \infty]$ denote the function

$$g_{ij}(\mathbf{x}_j, v) = \begin{cases} 0, & h_{ij}(\mathbf{x}_j) \le v \\ +\infty, & \text{otherwise} \end{cases} \ .$$

Then

$$(\mathbf{z}^{k+1}, \mathbf{v}^{k+1}) = \underset{(\mathbf{z}, \mathbf{v})}{\arg\min} \left\{ \sum_{i,j} \left[g_{ij}(\mathbf{z}_{ij}, v_{ij}) + \frac{\lambda}{2} \| (\mathbf{z}_{ij}, v_{ij}) - (\mathbf{x}_j^{k+1} + \frac{1}{\lambda}\mathbf{p}_{ij}^k, u_{ij}^{k+1} + \frac{1}{\lambda} q_{ij}^k) \|^2 \right] \right\} .$$

This minimization decomposes into md tasks

$$(\mathbf{z}_{ij}^k, v_{ij}^k) = \underset{\substack{\mathbf{z}_{ij} \in \mathbb{R}^{n_j} \\ u_{ij} \in \mathbb{R}}}{\arg\min} \{ g_{ij}(\mathbf{z}_{ij}, v_{ij}) + \frac{\lambda}{2} \| (\mathbf{z}_{ij}, v_{ij}) - (\mathbf{x}_j^{k+1} + \frac{1}{\lambda}\mathbf{p}_{ij}^k, u_{ij}^{k+1} + \frac{1}{\lambda} q_{ij}^k) \|^2 \} ,$$

that is, projecting $(\mathbf{x}_j^{k+1} + \frac{1}{\lambda}\mathbf{p}_{ij}^k, u_{ij}^{k+1} + \frac{1}{\lambda} q_{ij}^k)$ onto the *epigraph* of h_{ij} for $i = 1,...,m$ and $j=1,...,d$. We now consider general procedures for performing this kind of operation.

Proposition 5.4. Suppose $h: \mathbb{R}^l \to (-\infty, +\infty]$ is a closed proper convex function. Then the projection $(\mathbf{y}^*, v^*)$ of any $(\mathbf{x}, u) \in \mathbb{R}^l \times \mathbb{R}$ onto the set

$$\text{epi}(h) = \{(\mathbf{y}, v) \in \mathbb{R}^l \times \mathbb{R} \mid h(\mathbf{y}) \leq v\}$$

can be accomplished by:

$$\begin{aligned} \mathbf{y}^* &= \underset{\mathbf{y} \in \text{dom } h}{\arg\min} \left\{ \|\mathbf{y} - \mathbf{x}\|^2 + \max^2\{0, h(\mathbf{y}) - u\} \right\} \\ v^* &= \max \{u, h(\mathbf{y}^*)\} \quad . \end{aligned}$$

Proof. Let $\psi = \delta_{(\text{epi } h)}: \mathbb{R}^l \times \mathbb{R} \to (-\infty, +\infty]$. Then

$$(\mathbf{y}^*, v^*) = \underset{\mathbf{y}, v}{\arg\min} \left\{ \psi(\mathbf{y}, v) + \|\mathbf{y} - \mathbf{x}\|^2 + (v - u)^2 \right\} .$$

For any fixed $\mathbf{y} \in$ dom h, the minimum over v is attained at $v = u$ if $h(\mathbf{y}) \leq u$, or at $v = h(\mathbf{y})$ otherwise. For $\mathbf{y} \notin$ dom h, $\psi(\mathbf{y}, v) = \infty$ for all v. Therefore, minimizing first over v and then over $\mathbf{u}$, one obtains the indicated result. ■

Note that it would be advisable in practice to check whether $h(\mathbf{x}) \leq u$ at the outset, in which case $(\mathbf{x}, u) \in \text{epi } h$, and one can simply set $(\mathbf{y}^*, v^*) = (\mathbf{x}, u)$. Examples of the epigraphic projection calculation of Proposition 5.4 are depicted in Figures 14 and 15.

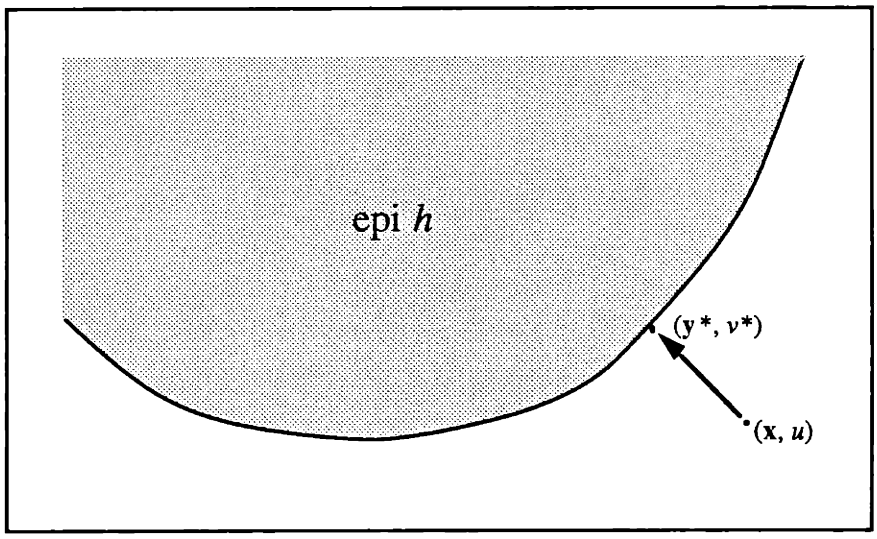


Figure 14: Example of projection onto the epigraph of a finite-valued convex function h.

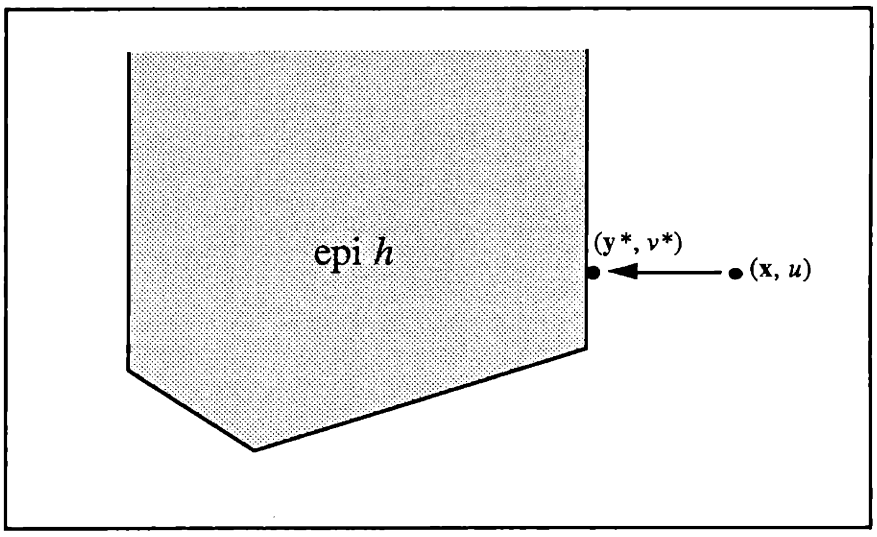


Figure 15: Projecting onto the epigraph of a convex function that can take the value $+\infty$.

In the case that h is convex and finite valued throughout $\mathbb{R}^l$, a simpler alternative to Proposition 5.4 is available.

Proposition 5.5. Suppose that $h: \mathbb{R}^l \to \mathbb{R}$ convex and everywhere finite. Then the projection $(\mathbf{y}^*, v^*)$ of any $(\mathbf{x}, u) \in \mathbb{R}^l \times \mathbb{R}$ onto the epi(h) can be accomplished by the following procedure:

```
if h(x) ≤ u
  then (y*, v*) := (x, u);
  else begin;
       y* := arg min_y {||y − x||² + (h(y) − u)²};
       v* := h(y*);
       end;
```

Proof. If $h(\mathbf{x}) \leq u$, then $(\mathbf{x}, u) \in$ epi(h), so $(\mathbf{y}^*, v^*) := (\mathbf{x}, u)$. Otherwise, $(\mathbf{x}, u) \notin$ epi(h). Being convex and finite, h is a continuous function, therefore $\eta(\mathbf{y}, v) = h(\mathbf{y}) - v$ is continuous. Now, epi $h = \{(\mathbf{y}, v) \mid \eta(\mathbf{y}, v) \leq 0\}$, so int(epi h) $= \{(\mathbf{y}, v) \mid \eta(\mathbf{y}, v) < 0\}$ and the boundary of epi h, bd(epi h), is $\{(\mathbf{y}, v) \mid \eta(\mathbf{y}, v) = 0\}$. The projection of $(\mathbf{x}, u)$ onto epi(h) must lie in bd(epi h), and therefore must be of the form $(\mathbf{y}, h(\mathbf{y}))$. The suggested procedure minimizes the distance between $(\mathbf{x}, u)$ and points of this form. ■

We let the notation epp(h, $\mathbf{x}$, u) stand for the projection of $(\mathbf{x}, u)$ onto epi h. The best procedure for calculating epp(h, $\mathbf{x}$, u) may be either that of Proposition 5.4 or that of Proposition 5.5, depending on whether or not h can take the value $+\infty$.

We return now to the issue of solving

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}, \mathbf{u}) + g(\mathbf{M}(\mathbf{x}, \mathbf{u})) \\ &\text{subject to} \quad (\mathbf{x}, \mathbf{u}) \in \mathbb{R}^n \times \mathbb{R}^{md} \quad . \end{aligned}$$

By combining the f minimization, the epigraphic projection procedures just discussed, and a straightforward multiplier update step, one obtains the method

$$\mathbf{x}_j^{k+1} = \arg\min_{\mathbf{x}_j \in \mathbb{R}^{N_j}} \{ h_{0j}(\mathbf{x}_j) + \langle \sum_{i=1}^{m} \mathbf{p}_{ij}^k, \mathbf{x}_j \rangle + \frac{m\lambda}{2} \| \mathbf{x}_j - \frac{1}{m} \sum_{i=1}^{m} \mathbf{z}_{ij}^k \|^2 \} , \qquad j=1,...,d$$

$$u_{ij}^{k+1} = \left(v_{ij}^k - \sum_{l=1}^{d} v_{il}^k \right) - \frac{1}{\lambda} \left(q_{ij}^k - \sum_{l=1}^{d} q_{il}^k \right), \qquad i = 1,...,m,\ j=1,...,d$$

$$(\mathbf{z}_{ij}^{k+1}, v_{ij}^{k+1}) = \mathrm{epp}(h_{ij}, \mathbf{x}_j^{k+1} + \tfrac{1}{\lambda}\mathbf{p}_{ij}^k, u_{ij}^{k+1} + \tfrac{1}{\lambda} q_{ij}^k), \qquad i = 1,...,m,\ j=1,...,d$$

$$\mathbf{p}_{ij}^{k+1} = \mathbf{p}_{ij}^k + \lambda(\mathbf{x}_j^{k+1} - \mathbf{z}_{ij}^{k+1}), \qquad i = 1,...,m,\ j=1,...,d$$

$$q_{ij}^{k+1} = q_{ij}^k + \lambda(u_j^{k+1} - v_{ij}^{k+1}), \qquad i = 1,...,m,\ j=1,...,d \ .$$

We call this algorithm the *epigraphic projection method* for block-separable convex programming. In appearance, it seems somewhat more complicated than Spingarn's method, and it uses more variables. However, in no part of the iteration does it have to minimize a function with a complicated structure such as

$$\chi_j(\mathbf{x}_j ; \mathbf{x}_j^k, \mathbf{u}, \mathbf{p}) = h_{0j}(\mathbf{x}_j) + \frac{\lambda}{2}\|\mathbf{x}_j - \mathbf{x}_j^k\|^2 + \frac{\lambda}{2} \sum_{i=1}^{m} \max^2\left\{0,\ h_{ij}(\mathbf{x}_j) - v_{ij}^k + \frac{1}{\lambda}\pi_i^k\right\} \ .$$

In fact, no minimizations performed by the method (including those inside the "epp" subroutine) involve more than one of the functions h_{ij} at a time. In the case where the h_{ij}, $i=1,...,m$, $j=1,...,d$ are everywhere finite-valued, one can use the method of Proposition 5.5 to implement the "epp" operation, the algorithm is not required to minimize any functions with "max" terms whatsoever. Thus, in practice, it might have sizable advantages of over Spingarn's method, which could conceivably have to minimize very complicated functions of the form χ_j.

In giving a convergence result for the epigraphic projection method, we will slightly relax the assumptions on the $\{h_{ij}\}$ made by Spingarn. We still require that the h_{ij} be convex

and everywhere finite-valued for $i = 1,...,m$, but for $i = 0$, let the h_{0j} be closed proper convex.

Proposition 5.6. Consider the problem (BSCP), where the h_{ij} are everywhere finite-valued convex for $i = 1,...,m$, and the h_{0j} are closed proper convex. Suppose that the following Slater condition is met: there exists some $\tilde{\mathbf{x}} \in \text{ri}(\text{dom } h_{01}) \times ... \times \text{ri}(\text{dom } h_{0d})$ such that $\sum_{j=1}^{d} h_{ij}(\tilde{\mathbf{x}}_j) < 0$ for $i = 1,...,m$. Then if (BSCP) is solvable, the sequence $\{\mathbf{x}^k\}$ produced by the epigraphic projection method will converge to a solution of (BSCP).

Proof. We will show that the problem

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}, \mathbf{u}) + g(\mathbf{M}(\mathbf{x}, \mathbf{u})) \\ \text{subject to} & (\mathbf{x}, \mathbf{u}) \in \mathbb{R}^n \times \mathbb{R}^{md} \end{array} \qquad \text{(PXU)}$$

possesses a Kuhn-Tucker pair. If (BSCP) is solvable, so is (P3), and hence (PXU). Let U denote the linear space

$$U = \{\mathbf{u} \in \mathbb{R}^{md} \mid \sum_{j=1}^{d} u_{ij} = 0,\ i = 1,...,m\} \ .$$

Then

$$\text{ri}(\text{dom } f) = \text{ri}(\text{dom } F_1) \times \text{ri}(\text{dom } F_2) = (\text{ri}(\text{dom } h_{01}) \times ... \times \text{ri}(\text{dom } h_{0d})) \times U \ .$$

For $i = 1,...,m$, let

$$r_i = -\sum_{j=1}^{d} h_{ij}(\tilde{\mathbf{x}}_j) > 0 \ ,$$

and define $\tilde{\mathbf{u}} \in \mathbb{R}^{md}$ via

$$\tilde{u}_{ij} = h_{ij}(\tilde{\mathbf{x}}_j) + \frac{r_i}{d}, \quad i = 1,...,m, j=1,...,d \ .$$

Then, for $i = 1,...,m$,

$$\sum_{j=1}^{d} \tilde{u}_{ij} \;=\; \sum_{j=1}^{d} h_{ij}(\tilde{\mathbf{x}}_j) \;+\; r_i \;=\; 0 \;,$$

and $\tilde{\mathbf{u}} \in U$. It follows that $(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \in \text{ri}(\text{dom}\, f)$.

Now, $\text{dom}\, g = \{(\mathbf{z}, \mathbf{u}) \mid h_{ij}(\mathbf{z}_{ij}) - u_{ij} \leq 0 \text{ for all } i, j\}$, and so

$$\text{dom}(g \circ \mathbf{M}) = \{(\mathbf{x}, \mathbf{u}) \mid h_{ij}(\mathbf{x}_j) - u_{ij} \;\leq\; 0,\, i = 1,...,m,\, j=1,...,d\} \;.$$

Now, the h_{ij}, $i = 1,...,m$, are finite and convex, hence continuous, so the functions

$$\eta_{ij}(\mathbf{x}_j, u_{ij}) = \; h_{ij}(\mathbf{x}_j) - u_{ij}$$

are also continuous. For all i and j,

$$\eta_{ij}(\tilde{\mathbf{x}}_j, \tilde{u}_{ij}) = \; h_{ij}(\tilde{\mathbf{x}}_j) - (h_{ij}(\tilde{\mathbf{x}}_j) \;+\; \frac{r_i}{d}) \;=\; -\frac{r_i}{d} < 0 \;.$$

It follows that there is an open neighborhood of $(\tilde{\mathbf{x}}, \tilde{\mathbf{u}})$ that is contained in $\text{dom}(g \circ \mathbf{M})$, and hence that

$$(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \;\in\; \text{int}(\text{dom}(g \circ \mathbf{M})) \;=\; \text{ri}(\text{dom}(g \circ \mathbf{M})) \;.$$

Therefore, $(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \;\in\; \text{ri}(\text{dom}\, f) \cap \text{ri}(\text{dom}(g \circ \mathbf{M}))$, and (PXU) is primal splittable, by Proposition 3.23. We also have, by similar reasoning,

$$\mathbf{M}(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) \;\in\; \text{int}(\text{dom}\, g) \;=\; \text{ri}(\text{dom}\, g) \;,$$

so (PXU) is also primal normal. By Proposition 3.31, the primal splittability, primal normality, and solvability of (PXU) imply that it possesses the Kuhn-Tucker pair $((\mathbf{x}^*, \mathbf{u}^*), (\mathbf{p}^*, \mathbf{q}^*))$. This existence is a sufficient condition for $\text{zer}(A+B) \neq \varnothing$, where

$A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$, and thus for the alternating direction method of multipliers with constant stepsize to converge. ■

We have thus shown that epigraphic projection converges under similar conditions to those needed for Spingarn's method. One could of course develop alternate qualification conditions assuring convergence by assuming that some of the functions h_{ij} are pseudo-polyhedral. Over/under-relaxed and approximate versions of the method are also possible, and can be derived by using the algorithm of Proposition 4.7 in place of (ADMOM$_\lambda$).

Interestingly, there is a method that shares many of the advantages of the epigraphic projection method over Spingarn's decomposition method, but can be derived using only the technique of partial inverses. The basic idea, essentially borrowed from Spingarn's method for a sum of S operators, is to rewrite (P3) as

$$\begin{array}{ll} \text{minimize} & F_1(\mathbf{x}) + F_2(\mathbf{u}) + F_3(\mathbf{z}, \mathbf{v}) \\ \text{subject to} & \mathbf{z} = \mathbf{H}\mathbf{x} \\ & \mathbf{u} = \mathbf{v} \quad , \end{array} \tag{P3'}$$

and then minimize $F_1(\mathbf{x}) + F_2(\mathbf{u}) + F_3(\mathbf{z}, \mathbf{v})$ over the subspace

$$W = \{(\mathbf{x}, \mathbf{u}, \mathbf{z}, \mathbf{v}) \mid \mathbf{z} = \mathbf{H}\mathbf{x}, \mathbf{u} = \mathbf{v}\} \ .$$

This can be done by letting $f(\mathbf{x}, \mathbf{u}, \mathbf{z}, \mathbf{v}) = F_1(\mathbf{x}) + F_2(\mathbf{u}) + F_3(\mathbf{z}, \mathbf{v})$ and $g = \delta_W$, and then applying the alternating direction method of multipliers. Equivalently, one can define f in the same manner and apply the proximal point algorithm to the partial inverse $(\partial f)_W$. In any case, the algorithm obtained should be similar to the epigraphic projection method, but not identical. We will not develop the theory of this alternative method any further.

Finally, the empirical properties of the epigraphic projection method, like those of Spingarn's, are currently unknown.

5.4. The Alternating Step Method for Monotropic Programming

We move now from block separable convex programs to problems with even more special structure — *monotropic programs*. Although monotropic programs are block-separable, we will not specialize the methods of the preceding two sections, but will instead apply the alternating direction method of multipliers in a slightly different way. First, we must define the term "monotropic program".

Definition 5.2 (Rockafellar 1984). A *monotropic program* is a convex program with a separable objective function, and only linear constraints. Its canonical form is

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} h_j(x_j) \\ \text{subject to} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^n \quad , \end{aligned} \tag{MP}$$

where $h_j: \mathbb{R} \to (-\infty, +\infty]$, $j=1,...,n$, are closed proper convex, $\mathbf{A}$ is an $m \times n$ matrix, and $\mathbf{b} \in \mathbb{R}^m$.

Since the h_j can take on the value $+\infty$, they may contain implicit constraints of the form $x_j \geq l_j \in \mathbb{R}$ and $x_j \leq u_j \in \mathbb{R}$. Thus, by proper choice of the h_j and the familiar device of slack variables, any separable convex optimization problem subject to linear equality and inequality constraints can be converted to form (MP).

Definition 5.3. Given a monotropic program (MP), define $d(i)$, the *degree of constraint* i, to be the number of nonzero elements in row i of $\mathbf{A}$.

If $\mathbf{A}$ is the node-arc incidence matrix of a network or graph, then this definition agrees with the usual notion of the degree of node i. Let $\mathbf{a}_j$ denote column j of $\mathbf{A}$, $1 \le j \le n$, and let $\mathbf{A}_i$ denote row i of $\mathbf{A}$, $1 \le i \le m$.

Definition 5.4. The *surplus* or *residual* $r_i(\mathbf{x})$ of constraint i of the system $\mathbf{Ax} = \mathbf{b}$, with respect to the primal variables $\mathbf{x}$, is $b_i - \langle \mathbf{A}_i, \mathbf{x} \rangle$. Let $\mathbf{r}(\mathbf{x})$ denote the vector $\mathbf{b} - \mathbf{Ax}$ of all m surpluses.

Intuitively, $r_i(\mathbf{x})$ is the amount by which the i^{th} constraint of the system $\mathbf{Ax} = \mathbf{b}$ is violated.

Let Q_i, for $1 \le i \le m$, be any set such that

$$\{ j \mid 1 \le j \le n, a_{ij} \ne 0 \} \subseteq Q_i \subseteq \{1,...,n\} \quad .$$

Let $q_i = |Q_i|$ for all i, hence $d(i) \le q_i \le n$. Furthermore, let

$$Q = \{(i, j) \mid 1 \le i \le m, j \in Q_i\} \quad .$$

We now convert (MP) to the form (P), minimize $f(\mathbf{x}) + g(\mathbf{Mx})$. Here, f will be defined on $\mathbb{R}^n$ and g on $\mathbb{R}^{mn}$. Index the components of vectors $\mathbf{z} \in \mathbb{R}^{mn}$ as z_{ij}, where $1 \le i \le m$ and $1 \le j \le n$. Then let

$$f(\mathbf{x}) = \sum_{j=1}^{n} h_j(x_j)$$

$$C = \{\, \mathbf{z} \in \mathbb{R}^{mn} \mid \sum_{j=1}^{n} z_{ij} = b_i \;\; \forall\, 1 \le i \le m,\;\; z_{ij} = 0 \;\; \forall\, (i,j) \notin Q \}$$

$$g(\mathbf{z}) \;=\; \delta_C(\mathbf{z}) \;=\; \begin{cases} 0, & \mathbf{z} \in C \\ +\infty, & \mathbf{z} \notin C \end{cases}$$

$$\mathbf{M} = \begin{bmatrix} \begin{bmatrix} \operatorname{diag}(\mathbf{A}_1) \end{bmatrix} \\ \begin{bmatrix} \operatorname{diag}(\mathbf{A}_2) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \operatorname{diag}(\mathbf{A}_m) \end{bmatrix} \end{bmatrix} ,$$

where, for any vector $\mathbf{v} \in \mathbb{R}^n$, diag($\mathbf{v}$) denotes the $n \times n$ matrix $\mathbf{D}$ with entries $d_{jj} = v_j$, j=1,...,n, along the diagonal, and zeroes elsewhere.

Proposition 5.7. With the above choices of f, g, and $\mathbf{M}$, the problem (P),

$$\text{minimize}_{\;\mathbf{x} \in \mathbb{R}^n} \;\; f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \quad ,$$

is equivalent to (MP) in the sense that $\mathbf{x}$ is feasible for (MP) if and only if $f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) < \infty$, and for all feasible $\mathbf{x}$, $f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) = \sum_{j=1}^{n} h_j(x_j)$.

Proof. A vector $\mathbf{x}$ is feasible for (MP) if and only if $h_j(x_j) < \infty$ for all j, and $\mathbf{A}\mathbf{x} = \mathbf{b}$. Now, $[\mathbf{M}\mathbf{x}]_{ij} = a_{ij}x_j$, and $a_{ij} = 0$ for all $(i,j) \notin Q$, so

$$g(\mathbf{Mx}) = \begin{cases} 0\ , & \sum_{j=1}^{n} a_{ij}x_j = b_i\ \forall\ i,\ \ a_{ij}x_j = 0\ \ \forall\ (i,j) \notin Q \\ +\infty\ , \text{otherwise} \end{cases}$$

$$= \begin{cases} 0\ , & \mathbf{Ax} = \mathbf{b} \\ +\infty\ , \text{otherwise} \end{cases} .$$

Thus, $f(\mathbf{x}) + g(\mathbf{Mx})$ is finite if and only if $\mathbf{x}$ is feasible for (MP), establishing the first claim. For the second claim, note that $\mathbf{x}$ being feasible implies $g(\mathbf{Mx}) = 0$, hence $f(\mathbf{x}) + g(\mathbf{Mx}) = f(\mathbf{x}) = \sum_{j=1}^{n} h_j(x_j)$. ■

Lemma 5.1. Unless $\mathbf{A}$ has a column that consists entirely of zeroes, the matrix $\mathbf{M}$ defined above has full column rank.

Proof. Let $\mathbf{x} \in \mathbb{R}^n$ be such that $\mathbf{Mx} = \mathbf{0}$. Then $a_{ij}x_j = 0$ for all i and j. As $\mathbf{A}$ is supposed to have no zero columns, there exists, for every j, an $i(j) \in \{1,..., m\}$ such that $a_{i(j)j} \neq 0$. It follows that $x_j = 0$ for all j, and that $\mathbf{x}$ is the zero vector. ■

For any $\mathbf{z} \in \mathbb{R}^{mn}$, and $1 \leq i \leq m$, let $\mathbf{z}_i \in \mathbb{R}^n$ be the subvector of $\mathbf{z}$ with components z_{ij}, j=1,...,n. Let

$$C_i = \{\ \mathbf{z}_i \in \mathbb{R}^n\ |\ \sum_{j=1}^{n} z_{ij} = b_i,\ z_{ij} = 0\ \ \forall\ j \notin Q_i\}\ ,$$

so that $C = C_1 \times ... \times C_m$. Note that the C_i are affine spaces, as is C.

We now apply the alternating direction method of multipliers to this formulation. As always, we use a fixed stepsize $\lambda > 0$.

The minimization

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x}\rangle + \frac{\lambda}{2}\|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2\}$$

decomposes into n independent computations

$$x_j^{k+1} = \arg\min_{x_j} \left\{ h_j(x_j) + \left(\sum_{i=1}^{m} p_{ij}^k a_{ij} \right) x_j + \frac{\lambda}{2} \sum_{i:\, j \in Q_i} (a_{ij}x_j - z_{ij}^k)^2 \right\} \quad ,$$

where $\mathbf{p}^k \in \mathbb{R}^{mn}$ is the multiplier vector, indexed in the same manner as $\mathbf{z}$. The minimization over $\mathbf{z}$,

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z}\rangle + \frac{\lambda}{2}\|\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}\|^2\} \quad ,$$

decomposes into the m problems

$$\mathbf{z}_i^{k+1} = \arg\min_{\mathbf{z}_i \in C_i} \left\{ -\langle \mathbf{p}_i^k, \mathbf{z}_i\rangle + \frac{\lambda}{2} \sum_{j=1}^{n} (a_{ij}x_j^{k+1} - z_{ij})^2 \right\}, \quad i=1,\ldots,m \quad .$$

To solve this problem for each i, we attach a single Lagrange multiplier π_i^{k+1} to the constraint $\sum_{j=1}^{n} z_{ij} = b_i$ which defines C_i. At the optimum, the Karush/Kuhn-Tucker conditions give, for all i and j such that $j \notin Q_i$, that

$$\nabla_{z_{ij}} \left\{ -\langle \mathbf{p}_i^k, \mathbf{z}_i\rangle + \pi_i^{k+1}(b_i - \sum_{j=1}^{n} z_{ij}) + \frac{\lambda}{2} \sum_{j=1}^{n} (a_{ij}x_j^{k+1} - z_{ij})^2 \right\} = 0 \quad .$$

Equivalently,

$$-p_{ij}^k - \pi_i^{k+1} + \lambda(z_{ij} - a_{ij}x_j^{k+1}) = 0 \qquad \forall\, j \in Q_i \ ,$$

and so we obtain that

$$z_{ij}^{k+1} = \begin{cases} a_{ij}x_j^{k+1} + \dfrac{p_{ij}^k + \pi_i^{k+1}}{\lambda} , & j \in Q_i \\ 0 , & j \notin Q_i \end{cases} .$$

We determine what π_i^{k+1} must be by setting

$$b_i = \sum_{j=1}^{n} z_{ij}^{k+1} = \sum_{j \in Q_i} \left(a_{ij}x_j^{k+1} + \frac{p_{ij}^k + \pi_i^{k+1}}{\lambda} \right)$$

$$\Rightarrow \quad b_i = \sum_{j \in Q_i} \left(a_{ij}x_j^{k+1} + \frac{p_{ij}^k}{\lambda} \right) + \frac{q_i}{\lambda}\pi_i^{k+1} \quad .$$

Rearranging the second equation, one obtains

$$\pi_i^{k+1} = \frac{\lambda}{q_i} \left(\left(b_i - \sum_{j \in Q_i} a_{ij}x_j^{k+1} \right) - \sum_{j \in Q_i} \frac{p_{ij}^k}{\lambda} \right)$$

$$\Rightarrow \quad \pi_i^{k+1} = \frac{1}{q_i} \left(\lambda r_i(\mathbf{x}^{k+1}) - \sum_{j \in Q_i} p_{ij}^k \right) \quad .$$

Finally, there is the update of the multipliers $\mathbf{p}$, which takes the form

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) \quad .$$

For (i, j) such that $(i, j) \in Q$, we have

$$\begin{aligned} p_{ij}^{k+1} &= p_{ij}^k + \lambda(a_{ij}x_j^{k+1} - z_{ij}^{k+1}) \\ &= p_{ij}^k + \lambda(a_{ij}x_j^{k+1} - a_{ij}x_j^{k+1} - \frac{1}{\lambda}[p_{ij}^k + \pi_i^{k+1}]) \\ &= -\pi_i^{k+1} \quad . \end{aligned}$$

For $(i, j) \notin Q$, the multiplier update formula implies $p_{ij}^{k+1} = p_{ij}^k$. That the $p_{ij}{}^{k+1}$, $(i, j) \in Q$, do not vary with j is by no means a coincidence, but is a consequence of $(\mathbf{p}^k, \mathbf{z}^k)$ being in $B = \partial g^*$ for all $k \geq 1$, by the construction of the alternating direction method of multipliers

(see Section 3.5.6). Here, ∂g is of the form N_C, where C is the affine subspace defined above. Letting V be the linear subspace parallel to C, that is

$$V = \{ \mathbf{z} \in \mathbb{R}^{mn} \mid \sum_{j=1}^{n} z_{ij} = 0, \forall 1 \le i \le m, z_{ij} = 0 \ \forall (i,j) \in Q \} \quad ,$$

we have from Proposition 3.5 that $\partial g^* = (\partial g)^{-1} = V^{\perp} \times C$, where $V^{\perp}$ is the orthogonal complement of V. It is easy to see that

$$V^{\perp} = \{ \mathbf{p} \in \mathbb{R}^{mn} \mid p_{ij} = p_{il} \ \forall (i,j), (i,l) \notin Q \} \quad .$$

Thus, it is no accident that the p_{ij}^{k+1}, $(i,j) \in Q$, do not vary with j.

In accordance with this observation, we make the assumption that $\mathbf{p}^0$ is such that for some $\pi^0 \in \mathbb{R}^m$, $p_{ij}^0 = -\pi_i^0$ for all $(i,j) \in Q$. It then follows $p_{ij}^k = -\pi_i^k$ for all $k \ge 0$ and $(i,j) \in Q$. The calculation of π_i^{k+1} now simplifies to

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda}{q_i} r_i(\mathbf{x}^{k+1}) \quad i = 1,...,m \quad ,$$

while that of z_{ij}^k, $(i,j) \in Q$, becomes, after simplification,

$$z_{ij}^k = a_{ij} x_j^{k+1} - \frac{1}{q_i} r_i(\mathbf{x}^{k+1}) \quad .$$

Now consider the "$\mathbf{x}$" minimization step,

$$x_j^{k+1} = \arg\min_{x_j} \left\{ h_j(x_j) + \left(\sum_{i=1}^{m} p_{ij}^k a_{ij} \right) x_j + \frac{\lambda}{2} \sum_{i:\, j \in Q_i} (a_{ij} x_j - z_{ij}^k)^2 \right\} \quad j = 1,...n \quad .$$

The values of p_{ij}^k, $(i,j) \notin Q$, are immaterial to this calculation because $a_{ij} = 0$ for all such (i,j). Thus, we can substitute $-\pi_i^k$ for p_{ij}^k throughout, and obtain the following implementation of the alternating direction method of multipliers:

$$x_j^{k+1} = \arg\min_{x_j} \left\{ h_j(x_j) - \left(\sum_{i=1}^{m} \pi_i^k a_{ij} \right) x_j + \frac{\lambda}{2} \sum_{i:\, j \in Q_i} (a_{ij} x_j - z_{ij}^k)^2 \right\}$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda}{q_i} r_i(\mathbf{x}^{k+1})$$

$$z_{ij}^{k+1} = \begin{cases} a_{ij} x_j^{k+1} - \frac{1}{q_i} r_i(\mathbf{x}^{k+1}), & (i,j) \in Q \\ 0, & (i,j) \notin Q \end{cases} .$$

Let us now assume that $\mathbf{z}^0$ is such that

$$z_{ij}^0 = \begin{cases} a_{ij} x_j^0 - \frac{1}{q_i} r_i(\mathbf{x}^0), & (i,j) \in Q \\ 0, & (i,j) \notin Q \end{cases}$$

for some $\mathbf{x}^0 \in \mathbb{R}^n$. Eliminating the z_{ij}^k and using that $(i,j) \notin Q$ implies $a_{ij} = 0$, we then get the method

$$x_j^{k+1} = \arg\min_{x_j} \left\{ h_j(x_j) - \langle \mathbf{a}_j, \pi^k \rangle x_j + \frac{\lambda}{2} \sum_{i=1}^{m} \left(a_{ij} x_j - \left(a_{ij} x_j^k + \frac{1}{q_i} r_i(\mathbf{x}^k) \right) \right)^2 \right\}$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda}{q_i} r_i(\mathbf{x}^{k+1}) \quad .$$

Collecting terms in the squared expressions, we obtain

$$x_j^{k+1} = \arg\min_{x_j} \left\{ h_j(x_j) - \langle \mathbf{a}_j, \pi^k \rangle x_j + \frac{\lambda \|\mathbf{a}_j\|^2}{2} \left(x_j - \left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2} \sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{x}^k)}{q_i} \right] \right)^2 \right\}$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda}{q_i} r_i(\mathbf{x}^{k+1}) \quad , \qquad \text{(ASMP)}$$

where $\mathbf{x}^0 \in \mathbb{R}^n$ and $\pi^0 \in \mathbb{R}^m$ may be chosen arbitrarily. Note that there is no longer any direct reference to the sets Q or Q_i. We merely allow q_i to be any integer between $d(i)$ and n, inclusive. In the degenerate case that $d(i)=0$ for some i, we must require that q_i be at least 1 so that (ASMP) remains well-defined. In practice, any such constraints of

degree zero can be immediately eliminated from the problem. Because the above algorithm (ASMP) involves alternating modifications of the primal variables x_j^k and the m dual variables π_i^k, we call it the *alternating step method* for monotropic programming.

Before stating the convergence properties of the alternating step method, we define a problem dual to (MP), namely

$$\text{minimize}_{\pi \in \mathbb{R}^m} \sum_{j=1}^{n} h_j^*(\mathbf{a}_j^T \pi) - \mathbf{b}^T \pi \quad , \tag{DMP}$$

where h_j^* denotes the convex conjugate of h_j, $j=1,...,n$. This definition is taken almost directly from Rockafellar (1984), Chapter 11A.

Theorem 5.1. Suppose one is given a monotropic program in the form (MP), where **A** has no column consisting entirely of zeroes. Assume also that h_j is pseudopolyhedral for all j, that is,

$$h_j(x) = \begin{cases} \bar{h}_j(x) , & x \in V_j \\ +\infty , & x \notin V_j \end{cases} ,$$

where each $\bar{h}_j$ is finite and convex throughout $\mathbb{R}$, and each $V_j \subseteq \mathbb{R}$ is a closed interval. Then if (MP) is solvable, $\{\mathbf{x}^k\}$ generated by the alternating step method converges to a solution of (MP), while $\{\pi^k\}$ converges to an optimal solution of the dual problem (DMP). Furthermore, in this case one also has $(x_j^*, \mathbf{a}_j^T \pi^*) \in \partial h_j$ for all j, where $\mathbf{x}^*$ and π^* are the respective limits of $\{\mathbf{x}^k\}$ and $\{\pi^k\}$. On the other hand, if (MP) is infeasible or unbounded, at least one of the sequences $\{\mathbf{x}^k\}$ and $\{\pi^k\}$ is unbounded.

Proof. Given the integers q_i, where $d(i) \le q_i \le n$ for $i = 1,...,m$, choose the sets Q_i in the definition of C and g above such that $|Q_i| = q_i$ for all i. We have already shown that (P),

with the above choices of f, g, and $\mathbf{M}$, is equivalent to (MP). Since $\mathbf{A}$ has no zero column, Lemma 5.1 gives that $\mathbf{M}$ has full column rank. Furthermore, by the analysis above, if we choose $p_{ij}^0 = -\pi_i^0$ for all i and j, and

$$z_{ij}^0 = \begin{cases} a_{ij}x_j^0 - \frac{1}{q_i} r_i(\mathbf{x}^0), & (i,j) \in Q \\ 0, & (i,j) \notin Q \end{cases},$$

then the sequences $\{\mathbf{x}^k\}$ evolved by (ADMOMλ) and (ASMP) are identical, and for the sequences $\{\mathbf{p}^k\}$ and $\{\mathbf{z}^k\}$, we have $p_{ij}^k = -\pi_i^k$ for all $k \geq 0$ and $(i,j) \in Q$, while for all $k \geq 0$,

$$z_{ij}^k = \begin{cases} a_{ij}x_j^k - \frac{1}{q_i} r_i(\mathbf{x}^k), & (i,j) \in Q \\ 0, & (i,j) \notin Q \end{cases}.$$

Since the h_j are all pseudopolyhedral, f is pseudopolyhedral. Now, g is a polyhedral function, hence pseudopolyhedral, so by the feasibility of (MP), (P) is primal splittable and normal. Since (MP) is solvable, so is (P), and (P) has a Kuhn-Tucker pair by Proposition 3.31. Thus, the sequences $\{\mathbf{x}^k\}$, $\{\mathbf{p}^k\}$, and $\{\mathbf{z}^k\}$ converge, by Proposition 3.42. In particular, $\{\mathbf{x}^k\}$ converges to a solution of (P), hence a solution of (MP). The convergence of $\{\mathbf{p}^k\}$ implies the convergence of $\{\pi^k\}$. Let

$$\mathbf{x}^* = \lim_{k \to \infty} \mathbf{x}^k \qquad \pi^* = \lim_{k \to \infty} \pi^k \quad .$$

Now $\mathbf{x}^*$ must be feasible, hence $\mathbf{A}\mathbf{x}^* = \mathbf{b}$, and $r_i(\mathbf{x}^*) = 0$ for all i. From

$$x_j^{k+1} = \arg\min_{x_j} \left\{ h_j(x_j) - \langle \mathbf{a}_j, \pi^k \rangle x_j + \frac{\lambda \|\mathbf{a}_j\|^2}{2} \left(x_j - \left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2} \sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{x}^k)}{q_i} \right] \right)^2 \right\}$$

we have, applying Rockafellar (1970a), Theorem 23.8, that for all j and k,

$$0 \in \partial h_j(x_j^{k+1}) - \mathbf{a}_j^T\pi^k + \lambda\|\mathbf{a}_j\|^2\left(x_j^{k+1} - \left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\sum_{i=1}^{m}\frac{a_{ij}r_i(\mathbf{x}^k)}{q_i}\right]\right) \quad .$$

In other words, for all j and k,

$$\left(x_j^{k+1},\ \mathbf{a}_j^T\pi^k - \lambda\|\mathbf{a}_j\|^2\left(x_j^{k+1} - \left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\sum_{i=1}^{m}\frac{a_{ij}r_i(\mathbf{x}^k)}{q_i}\right]\right)\right) \in \partial h_j \quad .$$

Taking limits and using Proposition 3.2,

$$\left(x_j^*,\ \mathbf{a}_j^T\pi^* - \lambda\|\mathbf{a}_j\|^2\left(x_j^* - \left[x_j^* + \frac{1}{\|\mathbf{a}_j\|^2}\sum_{i=1}^{m}\frac{a_{ij}r_i(x_j^*)}{q_i}\right]\right)\right) = (x_j^*,\ \mathbf{a}_j^T\pi^*) \in \partial h_j \quad .$$

As a result, $x_j^* \in \partial h_j^*(\mathbf{a}_j^T\pi^*)$ for all j. Let $h^*\colon \mathbb{R}^n \to (-\infty, +\infty]$ denote the function

$$h^*(\mathbf{x}) = \sum_{j=1}^{n} h_j^*(x_j) \quad ,$$

whence

$$\partial h^* = \bigotimes_{j=1}^{n} h_j^* \quad .$$

Then (DMP) can be written

$$\text{minimize } h^*(\mathbf{A}^T\pi) - \mathbf{b}^T\pi \quad .$$

From $x_j^* \in \partial h_j^*(\mathbf{a}_j^T\pi^*)$ for all j, we have $\mathbf{x}^* \in \partial h^*(\mathbf{A}^T\pi^*)$. Rockafellar (1970a), Theorem 23.9, gives $\mathbf{A}\mathbf{x}^* \in \partial[h^* \circ \mathbf{A}^T](\pi)$. Thus,

$$\partial_\pi\left[h^*(\mathbf{A}^T\pi) - \mathbf{b}^T\pi\right]_{\pi=\pi^*} \ni \mathbf{A}\mathbf{x}^* - \mathbf{b} = \mathbf{0} \quad ,$$

and π^* is optimal for (DMP). Here, the notation "∂_π" denotes the subgradient with respect to the variable π.

Lastly, we must also consider the case in which (MP) is not solvable. Then (P) cannot be solvable for the given choice of f, g, and $\mathbf{M}$, so Proposition 4.9 implies that at least one of the sequences $\{\mathbf{p}^k\}$ or $\{\mathbf{z}^k\}$ is unbounded. If $\{\mathbf{p}^k\}$ is unbounded, then $\{\pi^k\}$ is unbounded. If $\{\mathbf{z}^k\}$ is unbounded, we have from

$$z_{ij}^k = a_{ij}x_j^k - \frac{1}{q_i} r_i(\mathbf{x}^k), \quad k \geq 0, \ (i,j) \in Q,$$

that $\{\mathbf{x}^k\}$ must be unbounded. ■

The theorem does not cover the degenerate case in which $\mathbf{A}$ contains entirely zero columns. The variables x_j associated with such columns may be set to their optimal values by a simple one-dimensional minimizations of the corresponding h_j, and then removed from the problem.

Approximate and over/under-relaxed versions of the alternating step method can be derived using the generalized alternating direction method of multipliers of Proposition 4.7, as opposed to the basic alternating direction method of multipliers. The results are sufficiently interesting to warrant sketching the analysis. We recall the generalized alternating direction method of multipliers,

$$\text{Choose } \mathbf{x}^{k+1}\text{: } \left\| \mathbf{x}^{k+1} - \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle + \frac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z}^k \|^2 \right\} \right\| \leq \mu_k$$

$$\text{Choose } \mathbf{z}^{k+1}\text{: } \left\| \mathbf{z}^{k+1} - \arg\min_{\mathbf{z}} \left\{ g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z} \rangle + \frac{\lambda}{2} \| \rho_k \mathbf{M}\mathbf{x}^{k+1} + (1-\rho_k)\mathbf{z}^k - \mathbf{z} \|^2 \right\} \right\| \leq \omega_k$$

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda(\rho_k \mathbf{M}\mathbf{x}^{k+1} + (1-\rho_k)\mathbf{z}^k - \mathbf{z}^{k+1}) \quad .$$

In this case, the $\mathbf{z}$ minimization will be done exactly, that is, we will set $\omega_k = 0$ for all k, obtaining the slightly simpler method

$$\text{Choose } \mathbf{x}^{k+1}\text{: } \| \mathbf{x}^{k+1} - \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x}\rangle + \tfrac{\lambda}{2}\|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2\} \| \le \mu_k$$

$$\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z}\rangle + \tfrac{\lambda}{2}\|\rho_k\mathbf{M}\mathbf{x}^{k+1} + (1-\rho_k)\mathbf{z}^k - \mathbf{z}\|^2\}$$

$$\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda(\rho_k\mathbf{M}\mathbf{x}^{k+1} + (1-\rho_k)\mathbf{z}^k - \mathbf{z}^{k+1}) \quad .$$

We now apply this procedure to the current choice of f, g, and $\mathbf{M}$. The initial calculation, picking $\mathbf{x}^{k+1}$, can be implemented as an approximate version of the earlier one:

$$\text{Choose } x_j^{k+1}\text{: } \left\| x_j^{k+1} - \arg\min_{x_j} \left\{ h_j(x_j) + \left(\sum_{i=1}^{m} p_{ij}^k a_{ij} \right) x_j + \frac{\lambda}{2} \sum_{i:\, j \in Q_i} (a_{ij}x_j - z_{ij}^k)^2 \right\} \right\| \le \varepsilon_k \quad ,$$

where we let $\varepsilon_k = n^{(-1/2)}\mu_k$. For the calculation of $\mathbf{z}^{k+1}$, we now have $\rho_k\mathbf{M}\mathbf{x}^{k+1} + (1-\rho_k)\mathbf{z}^k$ in place of $\mathbf{M}\mathbf{x}^{k+1}$ in the quadratic term, but the problem decomposes much as before, yielding

$$\mathbf{z}_i^{k+1} = \arg\min_{\mathbf{z}_i \in C_i} \left\{ -\langle \mathbf{p}_i^k, \mathbf{z}_i\rangle + \frac{\lambda}{2} \sum_{j=1}^{n} (\rho_k a_{ij} x_j^{k+1} + (1-\rho_k) z_{ij}^k - z_{ij})^2 \right\}, \quad i=1,\ldots,m \quad .$$

Performing the same Lagrange multiplier analysis as before, we obtain

$$z_{ij}^{k+1} = \begin{cases} \rho_k a_{ij} x_j^{k+1} + (1-\rho_k) z_{ij}^k + \dfrac{p_{ij}^k + \pi_i^{k+1}}{\lambda} \ , & j \in Q_i \\ 0 \ , & j \notin Q_i \ . \end{cases}$$

We determine the values of the π_i^{k+1} by setting, for each i,

$$b_i = \sum_{j=1}^{n} z_{ij}^{k+1} = \rho_k \left(\sum_{j \in Q_i} a_{ij} x_j^{k+1} \right) + (1-\rho_k) \left(\sum_{j \in Q_i} a_{ij} z_{ij}^k \right) + \frac{1}{\lambda} \left(\sum_{j \in Q_i} p_{ij}^k + \pi_i^{k+1} \right) \ .$$

For all $k \ge 1$, $\sum_{j \in Q_i} a_{ij} z_{ij}^k = b_i$ by construction. Assume that $\mathbf{z}^0$ is chosen so that $\sum_{j \in Q_i} a_{ij} z_{ij}^0 = b_i$ for all i. Then, for all k, we can simplify the above equation to

$$\rho_k b_i = \rho_k \sum_{j \in Q_i} a_{ij} x_j^{k+1} + \frac{1}{\lambda} \sum_{j \in Q_i} \left(p_{ij}^k + \pi_i^{k+1} \right) .$$

Solving for π_i^{k+1},

$$\pi_i^{k+1} = \frac{1}{q_i} \left(\lambda \rho_k r_i(\mathbf{x}^{k+1}) - \sum_{j \in Q_i} p_{ij}^k \right) .$$

Just as before, and no more surprisingly, it then turns out that $p_{ij}{}^k = -\pi_i{}^k$ for all $k \geq 1$ and $(i, j) \in Q$. We further assume that there exists a π^0 such that $p_{ij}{}^0 = -\pi_i{}^0$ for every $(i, j) \in Q$. One can then substitute $-\pi_i{}^k$ for $p_{ij}{}^k$ everywhere, and obtain

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda \rho_k}{q_i} r_i(\mathbf{x}^{k+1})$$

$$z_{ij}^{k+1} = \begin{cases} \rho_k a_{ij} x_j^{k+1} + (1 - \rho_k) z_{ij}^k - \dfrac{\rho_k}{q_i} r_i(\mathbf{x}^{k+1}), & (i, j) \in Q \\ 0, & (i, j) \notin Q \end{cases} .$$

We then have the overall method

$$\text{Choose } x_j^{k+1}\text{: } \left\| x_j^{k+1} - \arg\min_{x_j} \left\{ h_j(x_j) - \langle \mathbf{a}_j, \pi^k \rangle x_j + \frac{\lambda}{2} \sum_{i: j \in Q_i} (a_{ij} x_j - z_{ij}^k)^2 \right\} \right\| \leq \varepsilon_k$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda \rho_k}{q_i} r_i(\mathbf{x}^{k+1})$$

$$z_{ij}^{k+1} = \begin{cases} \rho_k a_{ij} x_j^{k+1} + (1 - \rho_k) z_{ij}^k - \dfrac{1}{q_i} r_i(\mathbf{x}^{k+1}), & (i, j) \in Q \\ 0, & (i, j) \notin Q \end{cases} .$$

The elimination of the $\{\mathbf{z}^k\}$ is somewhat more subtle than before. Assume that $\mathbf{z}^0$ is of the form

$$z_{ij}^0 = \begin{cases} a_{ij}y_j^0 - \frac{1}{q_i} r_i(\mathbf{y}^0), & (i,j) \in Q \\ 0, & (i,j) \notin Q \end{cases}$$

for some $\mathbf{y}^0 \in \mathbb{R}^n$. Define the sequence $\{\mathbf{y}^k\}$ via the recursion

$$\mathbf{y}^{k+1} = (1-\rho_k)\mathbf{y}^k + \rho_k \mathbf{x}^{k+1} \quad \forall\, k \geq 0 \ .$$

We will now show that

$$z_{ij}^k = a_{ij}y_j^k - \frac{1}{q_i} r_i(\mathbf{y}^k) \quad \forall\, (i,j) \in Q, \ \forall\, k \geq 0 \ .$$

The statement holds by hypothesis for $k = 0$. Assuming it holds for k, we have, for all $(i,j) \in Q$,

$$\begin{aligned}
& a_{ij}y_j^{k+1} + \tfrac{1}{q_i} r_i(\mathbf{y}^{k+1}) \\
= \ & a_{ij}((1-\rho_k)y_j^k + \rho_k x_j^{k+1}) + \tfrac{1}{q_i}(b_i - \langle \mathbf{A}_i, \mathbf{y}^{k+1}\rangle) \\
= \ & (1-\rho_k)a_{ij}y_j^k + \rho_k a_{ij}x_j^{k+1} + \tfrac{1}{q_i}((1-\rho_k)(b_i - \langle \mathbf{A}_i, \mathbf{y}^k\rangle) + \rho_k(b_i - \langle \mathbf{A}_i, \mathbf{x}^{k+1}\rangle)) \\
= \ & (1-\rho_k)[a_{ij}y_j^k + \tfrac{1}{q_i} r_i(\mathbf{y}^k)] + \rho_k a_{ij}x_j^{k+1} + \tfrac{\rho_k}{q_i} r_i(\mathbf{x}^{k+1}) \\
= \ & (1-\rho_k)z_{ij}^k + \rho_k a_{ij}x_j^{k+1} + \tfrac{\rho_k}{q_i} r_i(\mathbf{x}^{k+1}) \\
= \ & z_{ij}^{k+1} \quad .
\end{aligned}$$

So, the claim holds by induction. We can now eliminate $\{\mathbf{z}^k\}$ to obtain the *generalized alternating step method* for monotropic programming:

$$\left\| x_j^{k+1} - \arg\min_{x_j} \left\{ h_j(x_j) - \langle \mathbf{a}_j, \pi^k \rangle x_j + \frac{\lambda \|\mathbf{a}_j\|^2}{2} \left(x_j - \left[y_j^k + \frac{1}{\|\mathbf{a}_j\|^2} \sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{y}^k)}{q_i} \right] \right)^2 \right\} \right\| \leq \varepsilon_k$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda \rho_k}{q_i} r_i(\mathbf{x}^{k+1})$$

$$\mathbf{y}^{k+1} = (1-\rho_k)\mathbf{y}^k + \rho_k \mathbf{x}^{k+1} \quad .$$

We now summarize this method's convergence properties:

Theorem 5.2. Let $\{\varepsilon_k\}_{k=0}^{\infty} \subseteq [0, \infty)$ and $\{\rho_k\}_{k=0}^{\infty} \subseteq (0, 2)$ be sequences such that

$$\sum_{k=0}^{\infty} \varepsilon_k < \infty \qquad 0 < \inf_{k \geq 0} \rho_k \leq \sup_{k \geq 0} \rho_k < 2 \quad .$$

Further suppose the monotropic program (MP) is solvable and that the h_j are pseudo-polyhedral for all j. Then for any initial $\pi^0 \in \mathbb{R}^m$ and $\mathbf{y}^0 \in \mathbb{R}^n$, the generalized alternating step method produces sequences $\{\mathbf{x}^k\}$ and $\{\pi^k\}$ converging to solutions $\mathbf{x}^*$ of (MP) and π^* the dual problem (DMP), respectively, with $(x_j^*, \mathbf{a}_j^T\pi^*) \in \partial h_j$ for all j.

Proof. The setup of (P) and the proof that (P) has a Kuhn-Tucker pair proceeds exactly as for Theorem 5.1. Letting $\mu_k = \sqrt{n}\varepsilon_k$ for all $k \geq 0$, which means that $\{\mu_k\}$ is summable, we have that the sequences $\{\mathbf{x}^k\}$, $\{\mathbf{z}^k\}$, and $\{\mathbf{p}^k\}$ evolved by the generalized alternating direction method of multipliers,

$$\begin{aligned}
&\text{Choose } \mathbf{x}^{k+1}\text{: } \left\| \mathbf{x}^{k+1} - \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle + \tfrac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z}^k \|^2 \right\} \right\| \leq \mu_k \\
&\mathbf{z}^{k+1} = \arg\min_{\mathbf{z}} \left\{ g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z} \rangle + \tfrac{\lambda}{2} \| \rho_k \mathbf{M}\mathbf{x}^{k+1} + (1 - \rho_k)\mathbf{z}^k - \mathbf{z} \|^2 \right\} \\
&\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda(\rho_k \mathbf{M}\mathbf{x}^{k+1} + (1 - \rho_k)\mathbf{z}^k - \mathbf{z}^{k+1}) \qquad ,
\end{aligned}$$

all converge. Let $\mathbf{x}^*$, $\mathbf{z}^*$, and $\mathbf{p}^*$ be the respective limits of these sequences. Then $\mathbf{x}^*$ is optimal for (P) and the equivalence of (MP) and (P) gives that $\mathbf{x}^*$ is optimal for (MP). The convergence of $\{\mathbf{p}^k\}$ implies the convergence of $\{\pi^k\}$ to some limit π^*. We now prove the dual optimality of π^*. Letting

$$\mathbf{s}^k = \mathbf{x}^{k+1} - \arg\min_{\mathbf{x}} \left\{ f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle + \tfrac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z}^k \|^2 \right\} \quad \forall\, k \geq 0 \ ,$$

whence $\mathbf{s}^k \to \mathbf{0}$, we have, for all k and j,

$$0 \in \partial h_j(x_j^{k+1} - s_j^k) - \mathbf{a}_j^T\pi^k + \lambda \sum_{i=1}^{n} a_{ij}\left(a_{ij}(x_j^{k+1} - s_j^k) - z_{ij}^k\right)$$

or equivalently

$$\left(x_j^{k+1} - s_j^k,\ \mathbf{a}_j^T\pi^k + \lambda \sum_{i=1}^{n} a_{ij}\left(a_{ij}(x_j^{k+1} - s_j^k) - z_{ij}^k\right)\right) \in \partial h_j \ .$$

Taking limits,

$$\left(x_j^*,\ \mathbf{a}_j^T\pi^* + \lambda \sum_{i=1}^{n} a_{ij}\left(a_{ij}x_j^* - z_{ij}^*\right)\right) \in \partial h_j \qquad j = 1,...,n \ .$$

Proposition 4.7 implies that $\mathbf{M}\mathbf{x}^* = \mathbf{z}^*$, hence $z_{ij}^* = a_{ij}x_j^*$ for all i and j, so $(x_j^*, \mathbf{a}_j^T\pi^*) \in \partial h_j$ for all j. This proves π^* is optimal for (DMP), in the same manner as in Theorem 5.1. ■

Notice that although $\{\mathbf{z}^k\}$ converges, no assertion is being made about the convergence of $\{\mathbf{y}^k\}$.

We could have derived the generalized alternating step method first, and then analyzed (ASMP) has a special case, but such an approach would have unnecessarily complicated the initial derivation.

Both the alternating step method and the generalized alternating step method are based on an alternating direction iterations in which some variables, namely $\{\mathbf{p}^k\}$ and $\{\mathbf{z}^k\}$, have dimension mn. However, both methods reduce to a form in which all the variables have dimension either m or n. A convergence proof involving only the lower-dimensional sequences $\{\mathbf{x}^k\} \in \mathbb{R}^n$, $\{\mathbf{y}^k\} \in \mathbb{R}^n$, $\{\pi^k\} \in \mathbb{R}^m$, and $\{\mathbf{r}(\mathbf{x}^k)\} = \{(r_1(\mathbf{x}^k),....r_m(\mathbf{x}^k))\} \in \mathbb{R}^m$, and also allowing the q_i to be non-integer, would be very appealing. At present, it does not appear that any such alternate proof exists, and the topic must be left open for future research.

As a final note, we should mention that an algorithm very similar to the alternating step method can be derived for block-separable convex programs subject only to linear inequality constraints, that is, problems of the form

$$\begin{aligned} &\text{minimize} && \sum_{j=1}^{d} h_j(\mathbf{x}_j) \\ &\text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{b} \quad , \end{aligned} \tag{BSLCP}$$

where h_j and $\mathbf{x}_j$ are defined as in Sections 5.2 and 5.3. For $1 \le i \le m$ and $1 \le j \le d$, let $\mathbf{A}_{ij}$ be the vector consisting of the elements a_{iq}, $q \in N_j$, where N_j is defined as in Section 5.2. Then, following a development similar to that of the alternating step method, one may derive the following specialization of the alternating direction method of multipliers for (BSLCP):

$$\begin{aligned} \mathbf{x}_j^{k+1} &= \arg\min_{\mathbf{x}_j \in \mathbb{R}^{N_j}} \left\{ h_j(\mathbf{x}_j) + \sum_{i:\, \mathbf{A}_{ij} \neq \mathbf{0}} \left[\pi_i^k \langle \mathbf{A}_{ij}, \mathbf{x}_j \rangle + \frac{\lambda}{2} \left(\langle \mathbf{A}_{ij}, \mathbf{x}_j - \mathbf{x}_j^k \rangle - \frac{r_i(\mathbf{x}^k)}{\hat{d}(i)} \right)^2 \right] \right\} \\ \pi_i^{k+1} &= \pi_i^k + \frac{\lambda}{\hat{d}(i)} r_i(\mathbf{x}^{k+1}) \quad . \end{aligned} \tag{BSLCM}$$

Here, the $\mathbf{x}_j^{k+1}$ calculation is done for $j=1,\ldots,d$, while the π_i^{k+1} calculation is done for $i=1,\ldots,m$. The quantities $r_i(\cdot)$ are defined as in the alternating step method. For $1 \le i \le m$, $\hat{d}(i)$ is defined to be the number of indices j for which $\mathbf{A}_{ij} \neq \mathbf{0}$. For a complete derivation in the case that the h_j are pseudopolyhedral, see Bertsekas and Tsitsiklis (1989), Section 3.4.4.

Chapter 6

The Alternating Step Method for Linear Programming

Monotropic programming contains, as perhaps its most important special case, the subject of linear programming. This chapter specializes the alternating step method of Section 5.4 to linear programming, obtaining a novel algorithm, the *alternating step method for linear programming*. This method is highly parallelizable, and has a number of features that make it different from conventional linear programming algorithms. We will show that the method need not converge in finite time, but that its convergence rate (at least for linear programs in standard primal form) must be at least linear. We will also study, from a theoretical point of view, possible parallel implementations of the method.

6.1. Deriving the Method

In this chapter, we will consider linear programs of the form

$$\begin{aligned} &\text{minimize} && \mathbf{c}^T\mathbf{x} \\ &\text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{b} \\ & && \mathbf{l} \le \mathbf{x} \le \mathbf{u} \quad , \end{aligned} \tag{LP}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{l} \in [-\infty, \infty)^n$, $\mathbf{u} \in (-\infty, \infty]^n$, and $\mathbf{A}$ is an $m \times n$ real matrix. This formulation is completely general, as it subsumes the standard primal form for which $l_j = 0$ and $u_j = \infty$ for $j=1,...,n$.

We further assume that $\mathbf{A}$ has no all-zero rows or columns. Neither of these assumptions is truly restrictive: all-zero rows either make the linear program infeasible (if the corres-

ponding b_i is non-zero), or can be immediately discarded (if the corresponding b_i is zero). Much as in general monotropic programming, all-zero columns can quickly be removed from the problem by setting the corresponding variables to their upper or lower bounds, depending on the signs of the corresponding cost coefficients.

For each $j=1,...,n$, define $V(j)$ to be the (possibly unbounded) real interval $[l_j, u_j] \cap \mathbb{R}$, the range of permissible values for x_j. then (LP) can be put into the standard monotropic programming form (MP) of Chapter 5 by setting

$$h_j(x_j) = \begin{cases} c_j x_j, & x_j \in V(j) \\ +\infty, & x_j \notin V(j) \end{cases}, \quad j=1,...,n \quad .$$

Let $\bar{c}_j(\pi)$ denote the reduced cost coefficient of the variable x_j with respect to the dual variables π, that is,

$$\bar{c}_j(\pi) = c_j - \pi^T \mathbf{a}_j \quad ,$$

and let $\bar{\mathbf{c}}(\pi)$ denote the vector $\mathbf{c} - \mathbf{A}^T\pi$ of these coefficients. Applying the alternating step method of monotropic programming,

$$\begin{aligned} x_j^{k+1} &= \arg\min_{x_j} \left\{ h_j(x_j) - \langle \mathbf{a}_j, \pi^k \rangle x_j + \frac{\lambda \|\mathbf{a}_j\|^2}{2} \left(x_j - \left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2} \sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{x}^k)}{q_i} \right] \right)^2 \right\} \\ \pi_i^{k+1} &= \pi_i^k + \frac{\lambda}{q_i} r_i(\mathbf{x}^{k+1}) \quad , \end{aligned} \tag{ASMP}$$

with the above choice of h_j, we obtain the method

$$\begin{aligned} x_j^{k+1} &= \arg\min_{x_j \in V(J)} \left\{ \bar{c}_j(\pi^k) x_j + \frac{\lambda \|\mathbf{a}_j\|^2}{2} \left(x_j - \left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2} \sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{x}^k)}{q_i} \right] \right)^2 \right\} \quad j=1,...,n \\ \pi_i^{k+1} &= \pi_i^k + \frac{\lambda}{q_i} r_i(\mathbf{x}^{k+1}) \quad , \qquad\qquad i=1,...,m \quad , \end{aligned}$$

where the q_i are any integers between $d(i)$ and n, inclusive, for $i=1,\ldots,m$ (the main reason for prohibiting zero rows is to keep the q_i from being zero). The minimand in the above calculation of x_j^{k+1} is quadratic, and so we may solve for x_j^{k+1} analytically. Setting the derivative to zero yields that the minimum over $x_j \in \mathbb{R}^n$ of

$$\bar{c}_j(\pi^k)x_j + \frac{\lambda\|\mathbf{a}_j\|^2}{2}\left(x_j - \left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\sum_{i=1}^{m}\frac{a_{ij}r_i(\mathbf{x}^k)}{q_i}\right]\right)^2$$

occurs at x_j such that

$$\bar{c}_j(\pi^k) + \lambda\|\mathbf{a}_j\|^2\left(x_j - \left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\sum_{i=1}^{m}\frac{a_{ij}r_i(\mathbf{x}^k)}{q_i}\right]\right) = 0$$

$$\Leftrightarrow \quad x_j = x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\left[\sum_{i=1}^{m}\frac{a_{ij}r_i(\mathbf{x}^k)}{q_i}\right] - \frac{\bar{c}_j(\pi^k)}{\lambda}\right) \quad .$$

To obtain the minimum over the interval $V(j)$, we must project this value of x_j onto $V(j)$, yielding the method

$$\boxed{\begin{aligned} x_j^{k+1} &= P_{V(j)}\left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\left[\sum_{i=1}^{m}\frac{a_{ij}r_i(\mathbf{x}^k)}{q_i}\right] - \frac{\bar{c}_j(\pi^k)}{\lambda}\right)\right] \qquad j=1,\ldots,n \\ \pi_i^{k+1} &= \pi_i^k + \frac{\lambda}{q_i}r_i(\mathbf{x}^{k+1}) \qquad\qquad i=1,\ldots,m \quad . \end{aligned}} \qquad \text{(ASLP)}$$

As in general monotropic programming, $\mathbf{x}^0 \in \mathbb{R}^n$ and $\pi^0 \in \mathbb{R}^m$ are arbitrary.

Theorem 6.1. Suppose that the matrix **A** of problem (LP) has no all-zero rows or columns. Then for any starting point $\mathbf{x}^0 \in \mathbb{R}^n$ and $\pi^0 \in \mathbb{R}^m$, and any integers $\{q_i\}_{k=1}^m$, where $d(i) \le q_i \le n$ for all i, the method (ASLP) produces a sequence $\{\mathbf{x}^k\}$ converging to

an optimal solution $\mathbf{x}^*$ of (LP), if such a solution exists. In this case, the sequence $\{\pi^k\}$ converges to a vector π^* of optimal simplex multipliers for (LP), that is

$$\begin{aligned} x_j^* > l_j \quad &\Rightarrow \quad \bar{c}_j(\pi^*) \le 0 \\ x_j^* < u_j \quad &\Rightarrow \quad \bar{c}_j(\pi^*) \ge 0 \quad . \end{aligned}$$

If, on the other hand, (LP) is infeasible or unbounded, at least one of the sequences $\{\mathbf{x}^k\}$ or $\{\pi^k\}$ is unbounded.

Proof. We have already shown that (ASLP) is equivalent to applying the alternating step method for monotropic programming to the problem (MP),

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n} h_j(x_j) \\ \text{subject to} \quad & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \in \mathbb{R}^n \quad , \end{aligned} \tag{MP}$$

where

$$h_j(x_j) = \begin{cases} c_j x_j, & l_j \le x_j \le u_j \\ +\infty, & \text{otherwise} \end{cases}, \quad j=1,\ldots,n \quad .$$

All the h_j are polyhedral, hence pseudopolyhedral. Therefore, by Theorem 5.1, if (MP) is solvable, that is, if (LP) is solvable, then $\mathbf{x}^k \to \mathbf{x}^*$ and $\pi^k \to \pi^*$, where $\mathbf{x}^*$ is optimal for (LP) and $(x_j^*, \mathbf{a}_j^T\pi) \in \partial h_j$ for all j. It remains to show that π^* is a vector of optimal multipliers. Calculating ∂h_j from the definition of h_j,

$$\partial h_j \;=\; \Big[\big((l_j, u_j) \times \{c_j\}\big) \cup \big(\{l_j\} \times (-\infty, c_j)\big) \cup \big(\{u_j\} \times (c_j, +\infty)\big)\Big] \cap \mathbb{R}^2 \quad .$$

See Figure 16 for a graphical depiction of ∂h_j. From this formula for ∂h_j, we see that if $x_j^* > l_j$, then $\mathbf{a}_j^T\pi \ge c_j$, that is, $\bar{c}_j(\pi^*) \le 0$. Likewise, if $x_j^* < u_j$, then $\mathbf{a}_j^T\pi \le c_j$, that is, $\bar{c}_j(\pi^*) \ge 0$. Finally, if (LP) is not solvable, then neither is the equivalent monotropic program (MP), and Theorem 5.1 implies that either $\{\mathbf{x}^k\}$ or $\{\pi^k\}$ is unbounded. ■

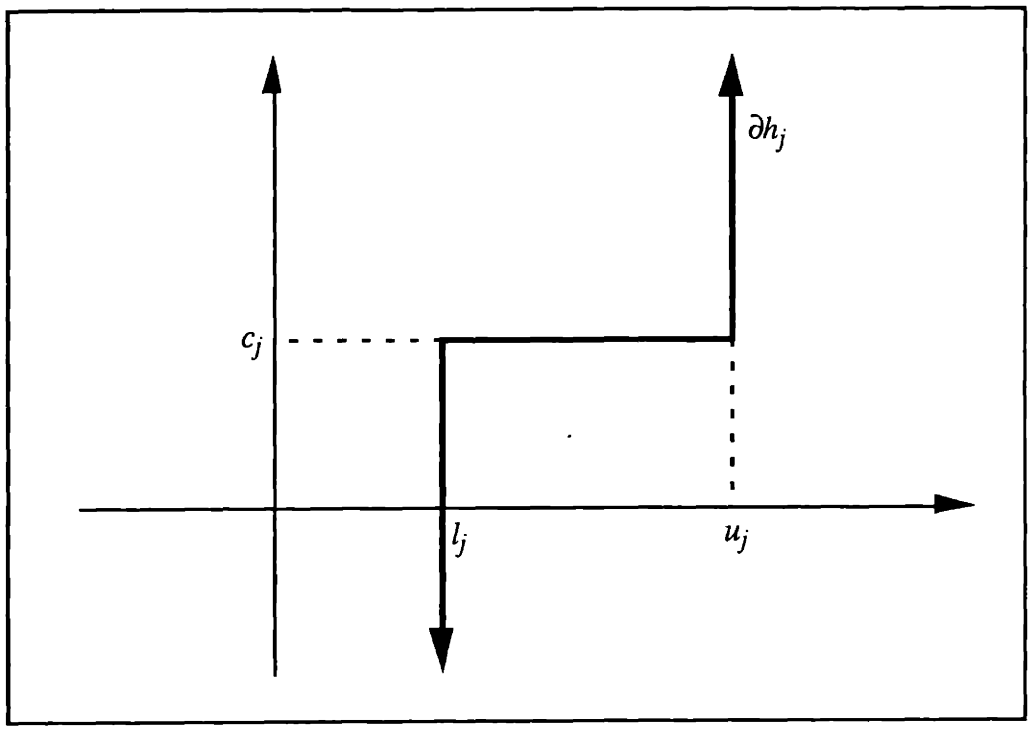


Figure 16. Graph of ∂h_j for the case in which l_j and u_j are both finite.

The alternating step method does not maintain any of the three customary invariants — primal feasibility, dual feasibility, and complementary slackness. Most existing linear programming algorithms preserve at least one of these properties while striving to satisfy the rest. This statement is also true of the new Karmarkar or interior point class of algorithms, which generally maintain primal feasibility at all times. By contrast, the alternating step method gradually works to satisfy all three properties as it progresses.

Some algorithms, such as the ϵ-relaxation and related methods for minimum cost network flow (Bertsekas 1985, 1986, Bertsekas and Eckstein 1987, 1988, Bertsekas and Tsitsiklis 1989, Goldberg 1987, Goldberg and Tarjan 1987) maintain a relaxed form of complementary slackness and dual feasibility. However, the alternating step method does not even maintain approximate versions of these properties. The only sense in which it may be

construed as preserving any form of feasibility or complementary slackness is that $\mathbf{x}^k$ always obeys the upper and lower bound constraints $\mathbf{l} \leq \mathbf{x} \leq \mathbf{u}$, and, in the higher-dimensional alternating direction iteration from which the algorithm is obtained,

$$x_j^{k+1} = \arg\min_{x_j} \left\{ h_j(x_j) + \left(\sum_{i=1}^{m} p_{ij}^k a_{ij} \right) x_j + \frac{\lambda}{2} \sum_{i:\, j \in Q_i} (a_{ij} x_j - z_{ij}^k)^2 \right\}$$

$$p_{ij}^{k+1} = p_{ij}^k - \frac{\lambda}{q_i} r_i(\mathbf{x}^{k+1})$$

$$z_{ij}^{k+1} = \begin{cases} a_{ij} x_j^{k+1} - \frac{1}{q_i} r_i(\mathbf{x}^{k+1}), & (i,j) \in Q \\ 0, & (i,j) \notin Q \end{cases},$$

the vectors $\{\mathbf{p}^k\}$ and $\{\mathbf{z}^k\}$ always satisfy $(\mathbf{z}^k, \mathbf{p}^k) \in N_C$, where C is the affine subspace defined in Section 5.4. The latter property is very different from classical complementary slackness for the problem (LP).

The alternating step method bears some resemblance to ε-relaxation and to the strictly convex network flow method of Bertsekas and El Baz (1987), in that the dual variables π_i are individually updated in response to the amount $r_i(\mathbf{x})$ to which the corresponding constraint is being violated. However, the dual update

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda}{q_i} r_i(\mathbf{x}^{k+1})$$

is unique in that the dual adjustment is *proportional* to the surplus $r_i(\mathbf{x}^{k+1})$, and the dual variable may either rise or fall, depending on the sign of the surplus. In the ε-relaxation and related algorithms, dual variables can be adjusted in only one direction.

To further illuminate the comparison between the alternating step method and ε-relaxation, consider the alternating step method specialized to minimum cost network

flow problems. If one assumes **A** is the node-arc incidence matrix of a network and uses the index vw, rather than j, to indicate arc (v, w), one obtains the method

$$x_{vw}^{k+1} = P_{V(vw)}\left[x_{vw}^{k} + \frac{1}{2}\left(\frac{r_v(\mathbf{x}^k)}{d(v)} - \frac{r_w(\mathbf{x}^k)}{d(w)} - \frac{\bar{c}_{vw}(\pi^k)}{\lambda}\right)\right] \qquad \forall \text{ arcs } (v, w)$$

$$\pi_v^{k+1} = \pi_v^k + \frac{\lambda}{d(v)} r_v(\mathbf{x}^{k+1}) \quad . \qquad \forall \text{ nodes } v \quad .$$

Here, x_{vw} denotes the flow on arc (v, w), π_v is the dual variable associated with node v, and $r_v(\mathbf{x})$ is the amount of flow imbalance at node v. We offer the following interpretation: each node v attempts to "push" away its surplus $r_v(\mathbf{x}^k)$, spreading it evenly among all its incident arcs. If the surplus is negative, the node instead "pulls" flow evenly on all its incident arcs. Each arc (v, w) then takes the two flows suggested by its "start" node v and "end" node w, averages them, makes a further adjustment based on its reduced cost, and then projects the result onto its feasible capacity range $V(vw)$. Each node then makes a price adjustment proportional to the surplus resulting from the new flows.

In summary, while there is some similarity to the the alternating "pushing" and price adjustment processes of ε-relaxation, the rules governing these steps are quite different. Furthermore, ε-relaxation has not been generalized to constraint structures more complicated than networks without gains, whereas the alternating step method may be applied to any monotropic program.

Another interesting feature of the alternating step method is that it does not solve a linear system of equations at each iteration, nor does it update a solution to such a system, as does the simplex method. However, by setting $c_j = 0$, $l_j = -\infty$, and $u_j = +\infty$ for all j, it could be used to solve the system of equations $\mathbf{Ax} = \mathbf{b}$. Thus, it is an iterative method that appears to be equally suited to optimizing linear programs or solving linear systems.

One can also apply the *generalized* alternating step method of Theorem 5.2,

$$\left\| x_j^{k+1} - \arg\min_{x_j} \left\{ h_j(x_j) - \langle \mathbf{a}_j, \pi^k \rangle x_j + \frac{\lambda \|\mathbf{a}_j\|^2}{2} \left(x_j - \left[y_j^k + \frac{1}{\|\mathbf{a}_j\|^2} \sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{y}^k)}{q_i} \right] \right)^2 \right\} \right\| \leq \varepsilon_k$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda \rho_k}{q_i} r_i(\mathbf{x}^{k+1})$$

$$\mathbf{y}^{k+1} = (1 - \rho_k)\mathbf{y}^k + \rho_k \mathbf{x}^{k+1} \quad ,$$

to linear programs in the form (LP), using the same setup as above. Since the minimizations to determine the x_j^{k+1} may be done analytically, the possibility of approximately computing the x_j^{k+1} is of little practical interest. Thus, we consider the special case of the generalized alternating step method in which $\varepsilon_k = 0$ for all k:

$$x_j^{k+1} = \arg\min_{x_j} \left\{ h_j(x_j) - \langle \mathbf{a}_j, \pi^k \rangle x_j + \frac{\lambda \|\mathbf{a}_j\|^2}{2} \left(x_j - \left[y_j^k + \frac{1}{\|\mathbf{a}_j\|^2} \sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{y}^k)}{q_i} \right] \right)^2 \right\}$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda \rho_k}{q_i} r_i(\mathbf{x}^{k+1})$$

$$\mathbf{y}^{k+1} = (1 - \rho_k)\mathbf{y}^k + \rho_k \mathbf{x}^{k+1} \quad .$$

Calculating the minima analytically, we obtain the *generalized alternating step method for linear programming*,

$$x_j^{k+1} = P_{V(j)}\left[y_j^k + \frac{1}{\|\mathbf{a}_j\|^2} \left(\left[\sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{y}^k)}{q_i} \right] - \frac{\bar{c}_j(\pi^k)}{\lambda} \right) \right] \qquad j=1,\ldots,n$$

$$y_j^{k+1} = (1 - \rho_k) y_j^k + \rho_k x_j^{k+1} \qquad j=1,\ldots,n \qquad \text{(GASLP)}$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda \rho_k}{q_i} r_i(\mathbf{x}^{k+1}) \qquad i=1,\ldots,m \quad .$$

From Theorem 5.2, we obtain

Theorem 6.2. Let $\{\rho_k\}_{k=0}^{\infty} \subseteq (0, 2)$ be a sequence such that

$$0 < \inf_{k \geq 0} \rho_k \leq \sup_{k \geq 0} \rho_k < 2 \quad ,$$

and suppose that the matrix **A** of the problem (LP) has no all-zero rows or columns. Further suppose that the $\{q_i\}_{k=1}^{m}$ are integers such that $d(i) \leq q_i \leq n$ for all i, and that (LP) is solvable. Then for any starting point $\mathbf{y}^0 \in \mathbb{R}^n$ and $\pi^0 \in \mathbb{R}^m$, the method (GASLP) produces a sequence $\{\mathbf{x}^k\}$ converging to an optimal solution $\mathbf{x}^*$ of (LP), and the sequence $\{\pi^k\}$ converging to a vector π^* of optimal simplex multipliers.

Proof. Again, we have that the h_j are pseudopolyhedral, so Theorem 5.2 implies that $\{\mathbf{x}^k\}$ and $\{\pi^k\}$ converge to respective limits $\mathbf{x}^*$ and π^* such that $(x_j^*, \mathbf{a}_j^T\pi) \in \partial h_j$ for all j. This proves the optimality of $\mathbf{x}^*$ and π^*, as in Theorem 6.1. ■

6.2. The Issue of Finite Convergence

In this section and the one following, we consider the rate at which the alternating step method converges. We begin by determining whether the convergence is finite, or merely asymptotic.

The alternating step method is equivalent to the alternating direction method of multipliers applied to f, g, and **M**, where

$$f(\mathbf{x}) = \begin{cases} \mathbf{c}^T\mathbf{x}, & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ +\infty & \text{otherwise} \end{cases} \quad ,$$

while g and **M** are as constructed in Section 5.4. Interestingly, if one were to apply the conventional method of multipliers,

$$(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}) = \arg\min_{\mathbf{x}, \mathbf{z}} \{f(\mathbf{x}) + g(\mathbf{z}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} - \mathbf{z} \rangle + \frac{\lambda}{2} \|\mathbf{M}\mathbf{x} - \mathbf{z}\|^2\}$$
$$\mathbf{p}^{k+1} = \mathbf{p}^k + \lambda(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) \quad ,$$

to the problem

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{x}) + g(\mathbf{z}) \\ &\text{subject to} \quad \mathbf{Mx} = \mathbf{z} \ , \end{aligned} \tag{P'}$$

it would always converge in a finite number of iterations. This behavior is due to the polyhedral character of f and g, and follows from a result originally proved by Bertsekas (1975), and strengthened by Luque (1984a). Because of this behavior, it is tempting to expect that the *alternating direction* method of multipliers might behave in a similar way.

We will first show why such an expectation is not reasonable, and then give an example of a linear program for which the alternating step method converges only asymptotically.

Fundamentally, the method of multipliers converges finitely on polyhedral problems because it corresponds to an application of the proximal point algorithm to a monotone operator that has a certain "staircase" property.

Definition 6.1. A monotone operator T on a Hilbert space $\mathcal{H}$ is said to be *staircase* if for all $y \in \text{im } T$, there exists some $\delta(y) > 0$ such that

$$w \in Tx, \ \|w - y\| < \delta(y) \quad \Rightarrow \quad y \in Tx \ .$$

T is called *locally staircase at zero* if such a condition holds for the single case $y = 0$, that is, there exists $\delta > 0$ such that

$$w \in Tx, \ \|w\| < \delta(y) \quad \Rightarrow \quad 0 \in Tx \ .$$

We use the term "staircase" because operators on $\mathbb{R}^1$ with the staircase property have graphs that resemble staircases (see Figure 17). Clearly, any staircase operator T with $0 \in \text{im } T$ is locally staircase at zero. The idea of a staircase operator is closely related to

the so-called "diff-max" property of convex functions (Durier 1986, Durier and Michelot 1986, Lefebvre and Michelot 1988). In brief, a convex function h is diff-max if and only if $(\partial h)^{-1}$ is staircase. In general, if the convex function h is polyhedral on $\mathbb{R}^n$, both ∂h and $(\partial h)^{-1} = \partial h^*$ are staircase (Durier 1986).

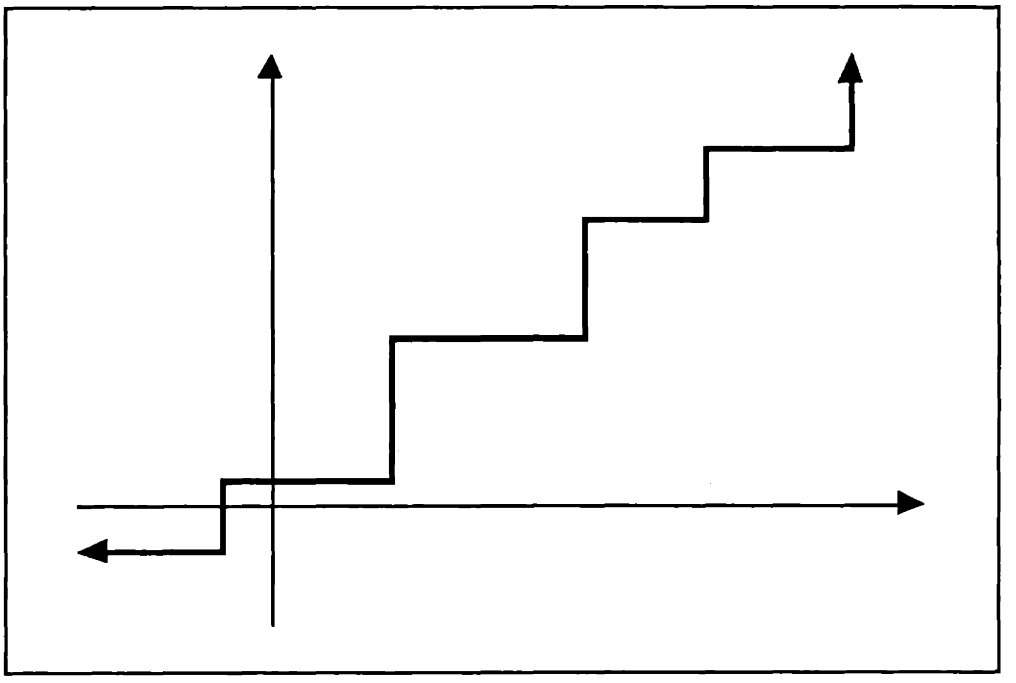


Figure 17. A staircase operator on $\mathbb{R}^1$.

Luque (1984a), building on earlier observations by Rockafellar (1976a), proved that the proximal point algorithm, when each iterate is computed exactly, converges finitely when applied to any operator T which is locally staircase at zero and has $\mathbf{0} \in \text{im } T$. The basic proof is very simple: suppose we have $\mathbf{x}^{k+1} = (I+\lambda T)^{-1}\mathbf{x}^k$ for all $k \geq 0$. Then $\frac{1}{\lambda}(\mathbf{x}^{k-1} - \mathbf{x}^k) \in T\mathbf{x}^k$ for all $k \geq 1$. For large enough k, we have $\| \frac{1}{\lambda}(\mathbf{x}^{k-1} - \mathbf{x}^k) \| < \delta$, implying $\mathbf{0} \in T\mathbf{x}^k$ and $\mathbf{x}^{k+1} = \mathbf{x}^k$.

Now, the alternating direction method of multipliers is the proximal point algorithm applied to the splitting operator $S_{\lambda,A,B}$, where $A = \partial[f^* \circ (-\mathbf{M}^T)]$ and $B = \partial g^*$. If $S_{\lambda,A,B}$ were to be staircase, then the alternating direction method of multipliers would converge finitely for any starting point. We will show, however, that $S_{\lambda,A,B}$ need not be staircase, even if both A and B are staircase.

Proposition 6.1. There exist maximal monotone operators A and B on $\mathbb{R}^n$, both staircase, such that $S_{\lambda,A,B}$ is not staircase for some choice of $\lambda > 0$.

Proof. We need to consider only the case of $\mathbb{R}^2$, and operators of the form $N_V = V \times V^\perp$, where V is a linear subspace. All operators of this form are staircase (in fact, for any $\mathbf{y} \in V^\perp$, $\delta(\mathbf{y})$ may be taken arbitrarily large). Define the following linear subspaces of $\mathbb{R}^2$:

$$\begin{aligned} W &= \{(x_1, x_2) \mid x_2 = 0\} = \{(x_1, 0) \mid x \in \mathbb{R}\} \\ U &= \{(x_1, x_2) \mid x_2 = \mu x_1\} = \{(x, \mu x) \mid x \in \mathbb{R}\} , \end{aligned}$$

where $\mu > 0$. Then

$$\begin{aligned} W^\perp &= \{(x_1, x_2) \mid x_1 = 0\} = \{(0, x_2) \mid x_2 \in \mathbb{R}\} \\ U^\perp &= \{(x_1, x_2) \mid x_2 = -\tfrac{1}{\mu} x_1\} = \{(z, -\tfrac{1}{\mu} z) \mid z \in \mathbb{R}\} . \end{aligned}$$

Following the analysis of Chapter 4,

$$\begin{aligned} S_{1,N_U,N_W} &= \{(\mathbf{x}_W - \mathbf{y}_{W^\perp}, \mathbf{y}_W - \mathbf{x}_{W^\perp}) \mid \mathbf{x} \in U, \mathbf{y} \in U^\perp\} \\ &= \{((x, \tfrac{1}{\mu} z), (z, -\mu x)) \mid x, z \in \mathbb{R}\} . \end{aligned}$$

Now, $S_{1,N_U,N_W}((x_1, x_2)) \ni (0, 0)$ if and only if $x_1 = x_2 = 0$. Thus S_{1,N_U,N_W} is not locally staircase at zero, and cannot be staircase. ■

The basic message of the above proposition is that forming splitting operators (or even partial inverses) can destroy the staircase property, and it therefore does not follow that the alternating direction method of multipliers will share the finite convergence properties of the conventional method of multipliers. This casts into doubt the prospects for finite convergence of the alternating step method, which is a special case of the alternating direction method of multipliers. In fact, we now prove:

Proposition 6.2. There exist solvable linear programs of the form (LP) such that for certain choices of $\lambda > 0$ and starting points $\mathbf{x}^0 \in \mathbb{R}^n$ and $\pi^0 \in \mathbb{R}^m$, the alternating step method does not converge finitely.

Proof. Again, we need look no farther than $\mathbb{R}^2$. Consider the (trivial) linear program

$$\begin{array}{lrcl} \min & 0x_1 + 0x_2 & & \\ \text{ST} & x_1 - x_2 & = & 0 \\ & x_2 & = & 0 \quad , \end{array}$$

where there are no finite bounds on x_1 or x_2. The only feasible solution is $x_1 = x_2 = 0$. We fix $\lambda = 1$. As may be seen by direct substitution into (ASLP), the alternating step method then produces iterates $x_1^k, x_2^k, \pi_1^k, \pi_2^k$ according to the following formulas:

$$\begin{aligned} x_1^{k+1} &= \tfrac{1}{2}x_1^k + \tfrac{1}{2}x_2^k + \pi_1^k \\ x_2^{k+1} &= \tfrac{1}{8}x_1^k + \tfrac{5}{8}x_2^k - \tfrac{1}{4}\pi_1^k + \tfrac{1}{4}\pi_2^k \end{aligned}$$

$$\begin{aligned} \pi_1^{k+1} &= \pi_1^k + \tfrac{1}{2}x_2^{k+1} - \tfrac{1}{2}x_1^{k+1} \\ \pi_2^{k+1} &= \pi_2^k - x_2^{k+1} \quad . \end{aligned}$$

Therefore

$$\begin{bmatrix} x_1^{k+1} \\ x_2^{k+1} \\ \pi_1^{k+1} \\ \pi_2^{k+1} \end{bmatrix} = \mathbf{W} \begin{bmatrix} x_1^{k} \\ x_2^{k} \\ \pi_1^{k} \\ \pi_2^{k} \end{bmatrix},$$

where

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 1 & 0 \\ \frac{1}{8} & \frac{5}{8} & -\frac{1}{4} & \frac{1}{4} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 1 & 0 \\ \frac{1}{8} & \frac{5}{8} & -\frac{1}{4} & \frac{1}{4} \\ -\frac{3}{16} & \frac{1}{16} & \frac{3}{8} & \frac{1}{8} \\ -\frac{1}{8} & -\frac{5}{8} & \frac{1}{4} & \frac{3}{4} \end{bmatrix}.$$

For (x_1, x_2, π_1, π_2) to be a fixed point of this iteration, one must have $x_1 = x_2 = 0$. From the update formulas above, it follows that one must also have $\pi_1 = \pi_2 = 0$. Thus, the iteration must converge to $(x_1, x_2, \pi_1, \pi_2) = (0, 0, 0, 0)$. We have by induction that

$$(x_1^k, x_2^k, \pi_1^k, \pi_2^k) = \mathbf{W}^k(x_1^0, x_2^0, \pi_1^0, \pi_2^0),$$

so if the iteration is to converge finitely from any starting point, then $\mathbf{W}$ must be nilpotent, that is, for some sufficiently large k, $\mathbf{W}^k$ is the zero matrix. However, expansion by cofactors gives that

$$\det(\mathbf{W}) = \det\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} \det\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 1 & 0 \\ \frac{1}{8} & \frac{5}{8} & -\frac{1}{4} & \frac{1}{4} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = 1 \cdot \det\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{8} & \frac{5}{8} \end{pmatrix} = \frac{1}{4},$$

so $\mathbf{W}$ is nonsingular, and cannot be nilpotent. ■

A recent article by Lefebvre and Michelot (1988) gives sufficient conditions for the method of partial inverses to converge finitely when applied to subgradient operators. It

would be interesting to extend these results to general operator splitting. However, Lefebvre and Michelot's sufficiency conditions appear to be too stringent to be of use in identifying nontrivial linear programs for which the alternating step method might be guaranteed to converge finitely.

6.3. Linear Convergence Rate

We now give a final theoretical result concerning the alternating step method: it may not converge finitely, but its convergence *rate* is linear. That is, the distance between $\mathbf{x}^k$ and the set of optimal solutions of the linear program decreases as a geometric sequence, as does the distance between π^k and the set of optimal simplex multipliers.

The basic line of analysis we will use is inspired by that of Luque (1984a), who, building on Rockafellar's (1976a) initial convergence rate results, gave conditions for the proximal point algorithm to converge in an asymptotically linear manner. However, our convergence rate analysis will be *global*, that is, the distance to the optimal set is bounded above, for all k, by a geometrically decreasing sequence.

6.3.1. Establishing a Connection with Linear Programming Stability Theory

In the ensuing analysis, we will make use of the l_∞-norm $\|\mathbf{w}\|_\infty = \max_j \{|w_j|\}$ and the l_1-norm $\|\mathbf{w}\|_1 = \sum_j \{|w_j|\}$, as well as the usual l_2-norm $\|\mathbf{w}\|_2 = \|\mathbf{w}\| = \left(\sum_j w_j{}^2\right)^{1/2}$.

Definition 6.2. We say that $\mathbf{x} \in \mathbb{R}^n$ is *δ-balanced* if $\mathbf{l} \le \mathbf{x} \le \mathbf{u}$ and $\|\mathbf{r}(\mathbf{x})\|_\infty \le \delta$, that is, if $r_i(\mathbf{x}^k) \le \delta$ for $i = 1,\ldots,m$. We also say that $(\mathbf{x}, \pi) \in \mathbb{R}^n \times \mathbb{R}^m$ obey *ε-complementary slackness* if $\mathbf{l} \le \mathbf{x} \le \mathbf{u}$ and

$$x_j > l_j \quad \Rightarrow \quad \bar{c}_j(\pi) \leq \varepsilon$$
$$x_j < u_j \quad \Rightarrow \quad \bar{c}_j(\pi) \geq -\varepsilon \ .$$

Alternatively, $\mathbf{x}$ is δ-balanced if and only if it is feasible for some problem obtained from (LP) by replacing $\mathbf{b}$ by some $\mathbf{b}'$, where $\| \mathbf{b} - \mathbf{b}' \|_\infty \leq \delta$, and $(\mathbf{x}, \pi)$ obey ε-complementary slackness if and only if they obey conventional complementary slackness with respect to some costs $\mathbf{c}'$, where $\| \mathbf{c} - \mathbf{c}' \|_\infty \leq \varepsilon$.

We recall that the alternating step method is equivalent to the alternating direction method of multipliers

$$\begin{aligned} \mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} \{ f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle + \tfrac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z}^k \|^2 \} \\ \mathbf{z}^{k+1} &= \arg\min_{\mathbf{z}} \{ g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z} \rangle + \tfrac{\lambda}{2} \| \mathbf{M}\mathbf{x}^{k+1} - \mathbf{z} \|^2 \} \qquad (\text{ADMOM}_\lambda) \\ \mathbf{p}^{k+1} &= \mathbf{p}^k + \lambda (\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) \end{aligned}$$

as applied to

$$f(\mathbf{x}) = \begin{cases} \mathbf{c}^T\mathbf{x}, & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ +\infty & \text{otherwise} \end{cases}$$

$$g(\mathbf{z}) = \delta_C(\mathbf{z}) = \begin{cases} 0, & \mathbf{z} \in C \\ +\infty, & \mathbf{z} \notin C \end{cases} ,$$

where

$$C = \{ \mathbf{z} \in \mathbb{R}^{mn} \mid \sum_{j=1}^{n} z_{ij} = b_i \ \forall 1 \leq i \leq m, \ z_{ij} = 0 \ \forall (i,j) \notin Q \} \ ,$$

and

$$\mathbf{M} = \begin{bmatrix} \begin{bmatrix} \operatorname{diag}(\mathbf{A}_1) \end{bmatrix} \\ \begin{bmatrix} \operatorname{diag}(\mathbf{A}_2) \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \operatorname{diag}(\mathbf{A}_m) \end{bmatrix} \end{bmatrix} .$$

Now, (ADMOM$_\lambda$) is itself equivalent to the proximal point algorithm with constant stepsize 1, as applied to the maximal monotone operator $S_{\lambda,A,B}$, where

$$A = \partial[f^* \circ (-\mathbf{M}^T)] = -\mathbf{M} \circ (\partial f)^{-1} \circ (-\mathbf{M}^T)$$
$$B = \partial g^* = (\partial g)^{-1} .$$

Note that we can write $\partial[f^* \circ (-\mathbf{M}^T)] = -\mathbf{M} \circ \partial f^* \circ (-\mathbf{M}^T)$ because $\operatorname{im}(-\mathbf{M}^T) = \mathbb{R}^n$. We now give explicit expressions for A and B. First,

$$\partial f(\mathbf{x}) = \begin{cases} \{\mathbf{c}+\xi \in \mathbb{R}^n \mid x_j < u_j \Rightarrow \xi_j \le 0,\ x_j > l_j \Rightarrow \xi_j \ge 0\ \}, & \mathbf{l} \le \mathbf{x} \le \mathbf{u} \\ \varnothing, & \text{otherwise} . \end{cases}$$

Thus,

$$(\partial f)^{-1}(\mathbf{y}) = \{\ \mathbf{x} \in \mathbb{R}^n \mid \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ y_j < c_j \Rightarrow x_j = l_j,\ y_j > c_j \Rightarrow x_j = u_j\} ,$$

and so for any $\mathbf{q} \in \mathbb{R}^{mn}$,

$$A\mathbf{q} = -\mathbf{M}(\partial f)^{-1}(-\mathbf{M}^{\mathrm{T}}\mathbf{q})$$
$$= \left\{ [-a_{ij}x_j] \in \mathbb{R}^{mn} \,\middle|\, \mathbf{x} \in \mathbb{R}^n, \mathbf{l} \le \mathbf{x} \le \mathbf{u},\ x_j = l_j \ \forall j\colon c_j + \sum_{i=1}^{m} a_{ij}q_{ij} > 0,\right.$$
$$\left. x_j = u_j \ \forall j\colon c_j + \sum_{i=1}^{m} a_{ij}q_{ij} < 0 \right\} .$$

Note that this definition implies that $A\mathbf{q}=\varnothing$ if there exists some j such that

$$l_j = -\infty \text{ and } c_j + \sum_{i=1}^{m} a_{ij}q_{ij} > 0 ,$$

or

$$u_j = +\infty \text{ and } c_j + \sum_{i=1}^{m} a_{ij}q_{ij} < 0 .$$

The expression for $B = (\partial g)^{-1}$ is much simpler. As in Section 5.4, we have $\partial g = C \times V^{\perp}$, where $V^{\perp}$ is the orthogonal complement of the linear subspace V parallel to C, namely

$$V^{\perp} = \{ \mathbf{p} \in \mathbb{R}^{mn} \mid p_{ij} = p_{il} \ \forall (i,j), (i,l) \in Q \} .$$

It then follows that for any $\mathbf{p} \in V^{\perp}$, $B\mathbf{p} = C$, and otherwise $B\mathbf{p}=\varnothing$.

We now state a technical lemma:

Lemma 6.1. Suppose $(\mathbf{t}, \mathbf{w}) \in S_{\lambda,A,B}$. Then there exist $\mathbf{p}, \mathbf{q}, \mathbf{s}, \mathbf{z} \in \mathbb{R}^{mn}$, $\mathbf{x} \in \mathbb{R}^n$, and $\pi \in \mathbb{R}^m$ such that

$$\begin{aligned}
&(\mathbf{p}, \mathbf{z}) \in B \\
&(\mathbf{q}, \mathbf{s}) \in A \\
&\mathbf{q} + \lambda\mathbf{s} = \mathbf{p} - \lambda\mathbf{z} \\
&\mathbf{q} + \lambda\mathbf{z} = \mathbf{t} \\
&\mathbf{p} - \mathbf{q} = \mathbf{w} \\
&s_{ij} = -a_{ij}x_j \quad \text{for } i = 1,\ldots,m,\ j=1,\ldots,n \\
&\pi_i = -p_{ij} \quad \text{for all } (i,j) \in Q ,
\end{aligned}$$

and for any $\mathbf{p}$, $\mathbf{q}$, $\mathbf{s}$, $\mathbf{z}$, $\mathbf{x}$ and π satisfying these conditions,

$\mathbf{x}$ is $d_{\max}\|\mathbf{w}\|/\lambda$-balanced

$(\mathbf{x}, \pi)$ obeys $a_{\max}\|\mathbf{w}\|$-complementary slackness,

where $a_{\max} \triangleq \max_{j=1,\ldots,n} \{\|\mathbf{a}_j\|\}$ and $d_{\max} \triangleq \max_{i=1,\ldots,m} \{d(i)\}$.

Proof. From the definition of $S_{\lambda,A,B}$,

$$S_{\lambda,A,B} = \{(\mathbf{q} + \lambda\mathbf{z}, \mathbf{p} - \mathbf{q}) \mid (\mathbf{p}, \mathbf{z}) \in B, (\mathbf{q}, \mathbf{s}) \in A, \mathbf{q}+\lambda\mathbf{s} = \mathbf{p} - \lambda\mathbf{z}\} ,$$

we immediately have that the existence of $\mathbf{p}, \mathbf{q}, \mathbf{s}, \mathbf{z} \in \mathbb{R}^{mn}$ satisfying the first five conditions,

$$\begin{aligned}
&(\mathbf{p}, \mathbf{z}) \in B \\
&(\mathbf{q}, \mathbf{s}) \in A \\
&\mathbf{q}+\lambda\mathbf{s} = \mathbf{p} - \lambda\mathbf{z} \\
&\mathbf{q}+\lambda\mathbf{z} = \mathbf{t} \\
&\mathbf{p} - \mathbf{q} = \mathbf{w} .
\end{aligned}$$

In accordance with $(\mathbf{q}, \mathbf{s}) \in A$ and the form of A, let $\mathbf{x}' \in \mathbb{R}^n$ be such that $\mathbf{x}'$ obeys the bounds $\mathbf{l} \le \mathbf{x}' \le \mathbf{u}$, $x_j' = l_j$ for all j such that $c_j + \sum_{i=1}^{m} a_{ij}q_{ij} > 0$, $x_j' = u_j$ for all j such that $c_j + \sum_{i=1}^{m} a_{ij}q_{ij} < 0$, and $s_{ij} = - a_{ij}x_j'$ for all i and j. Given any $\mathbf{x}$ with the property $s_{ij} = - a_{ij}x_j$ for all i and j, the fact that $\mathbf{A}$ has no zero columns gives $\mathbf{x}' = \mathbf{x}$. Note that $\mathbf{q}+\lambda\mathbf{s} = \mathbf{p} - \lambda\mathbf{z}$ implies $\lambda(\mathbf{s} + \mathbf{z}) = \mathbf{p} - \mathbf{q} = \mathbf{w}$. We have

$$[\mathbf{s} + \mathbf{z}]_{ij} = z_{ij} - a_{ij}x_j \quad \forall\ i, j ,$$

and, as $\|\mathbf{s} + \mathbf{z}\| = \|\mathbf{w}\|/\lambda$,

$$| z_{ij} - a_{ij}x_j | \le \|\mathbf{w}\|/\lambda \quad \forall\ i, j ,$$

whence

$$\Big| \sum_{j:\, a_{ij} \neq 0} (z_{ij} - a_{ij}x_j) \Big| \le d(i)\|\mathbf{w}\|/\lambda \quad \forall\ i .$$

Since $(\mathbf{p}, \mathbf{z}) \in B = V^{\perp} \times C$, we must have $\mathbf{z} \in C$, and so $\sum_{j=1}^{n} z_{ij} = b_i$ for all i. Thus, the last inequality is equivalent to

$$| b_i - \mathbf{A}_i^{\mathsf{T}} \mathbf{x} | = | r_i(\mathbf{x}) | \leq d(i) \|\mathbf{w}\| / \lambda \quad \forall\, i\,.$$

Thus, using the definition of $d_{\max}$, we conclude that the primal solution $\mathbf{x}$ is $d_{\max}\|\mathbf{w}\|/\lambda$-balanced.

From $(\mathbf{p}, \mathbf{z}) \in B = V^{\perp} \times C$, we also have $\mathbf{p} \in V^{\perp}$, hence $p_{ij} = p_{il}$ for all $(i, j), (i, l) \in Q$. In particular, $p_{ij} = p_{il}$ for all $(i, j), (i, l)$ such that $a_{ij}, a_{il} \neq 0$. Then it is possible to uniquely define $\pi \in \mathbb{R}^m$ via $\pi_i' = -p_{ij}$ for all (i, j) such that $a_{ij} \neq 0$. We now consider $\mathbf{p}$ and $\mathbf{q}$. For all j,

$$\left| \sum_{i=1}^{m} a_{ij}(p_{ij} - q_{ij}) \right| = | \mathbf{a}_j^{\mathsf{T}}(\pi + \mathbf{q}_j) | \leq \|\mathbf{a}_j\| \cdot \|\pi + \mathbf{q}_j\|\,,$$

where $\mathbf{q}_j$ is them m-vector consisting of the $\{q_{ij}\}_{i=1}^{m}$. Since $\|\mathbf{p} - \mathbf{q}\| = \|\mathbf{w}\|$, $\|\pi + \mathbf{q}_j\| \leq \|\mathbf{w}\|$, and thus

$$\left| \sum_{i=1}^{m} a_{ij}(p_{ij} - q_{ij}) \right| \leq \|\mathbf{a}_j\| \cdot \|\mathbf{w}\|\,.$$

Using that $(\mathbf{q}, \mathbf{s}) \in A$, we have

$$x_j < u_j \;\Rightarrow\; c_j + \sum_{i=1}^{m} a_{ij} q_{ij} \geq 0 \;\Leftrightarrow\; \left(c_j - \sum_{i=1}^{m} a_{ij}(p_{ij} - q_{ij})\right) - \sum_{i=1}^{m} a_{ij}\pi_i \geq 0$$

$$x_j > l_j \;\Rightarrow\; c_j + \sum_{i=1}^{m} a_{ij} q_{ij} \leq 0 \;\Leftrightarrow\; \left(c_j - \sum_{i=1}^{m} a_{ij}(p_{ij} - q_{ij})\right) - \sum_{i=1}^{m} a_{ij}\pi_i \leq 0\,.$$

Thus, $(\mathbf{x}, \pi)$ obey complementary slackness if we permit a perturbation of each cost coefficient c_j by an amount $\sum_{i=1}^{m} a_{ij}(p_{ij} - q_{ij})$, whose magnitude cannot exceed $\|\mathbf{a}_j\| \cdot \|\mathbf{w}\|$. Using that $a_{\max} = \max_j \{\|\mathbf{a}_j\|\}$, we conclude $(\mathbf{x}, \pi)$ obeys $a_{\max}\|\mathbf{w}\|$-complementary slackness. ■

We now precisely characterize the zeroes of the splitting operator.

Lemma 6.2. For the choices of the maximal monotone operators A and B used in the alternating step method for linear programming,

$$\begin{aligned}\mathrm{zer}(S_{\lambda,A,B}) = \{\, \zeta \in \mathbb{R}^{mn} \mid\ & \zeta_{ij} = -\pi_i + \lambda a_{ij} x_j \ \forall\ (i,j) \in Q, \\ & \mathbf{x} \text{ is an optimum of (LP)} \\ & \pi \text{ is a vector of optimal simplex multipliers for (LP)} \,\}\ .\end{aligned}$$

Proof. Using Proposition 4.3,

$$\mathrm{zer}(S_{\lambda,A,B}) = \{\mathbf{p}+\lambda\mathbf{z} \mid (\mathbf{p}, \mathbf{z}) \in B, (\mathbf{p}, -\mathbf{z}) \in A\}$$

$$\begin{aligned}= \{\, [p_{ij} + \lambda a_{ij} x_j] \mid\ & p_{ij} = p_{il} \ \forall\ (i,j), (i,l) \in Q, \\ & \mathbf{x} \in \mathbb{R}^n, \mathbf{l} \le \mathbf{x} \le \mathbf{u}, \\ & \textstyle\sum_{j=1}^{n} a_{ij} x_j = b_i \ \ \forall i, \\ & x_j = l_j \ \forall j\colon c_j + \textstyle\sum_{i=1}^{m} a_{ij} p_{ij} > 0, \\ & x_j = u_j \ \forall j\colon c_j + \textstyle\sum_{i=1}^{m} a_{ij} p_{ij} < 0 \,\}\end{aligned}$$

$$\begin{aligned}= \{\, \zeta \in \mathbb{R}^{mn} \mid\ & \zeta_{ij} = -\pi_i + \lambda a_{ij} x_j \ \forall\ (i,j) \in Q, \\ & \pi \in \mathbb{R}^m, \mathbf{x} \in \mathbb{R}^n, \mathbf{l} \le \mathbf{x} \le \mathbf{u}, \\ & \textstyle\sum_{j=1}^{n} a_{ij} x_j = b_i \ \ \forall i, \\ & x_j = l_j \ \forall j\colon\ \bar{c}_j(\pi) > 0, \\ & x_j = u_j \ \forall j\colon\ \bar{c}_j(\pi) < 0 \,\}\end{aligned}$$

$$\begin{aligned}= \{\, \zeta \in \mathbb{R}^{mn} \mid\ & \zeta_{ij} = -\pi_i + \lambda a_{ij} x_j \ \forall\ (i,j) \in Q, \\ & \mathbf{x} \text{ is an optimum of (LP)} \\ & \pi \text{ is a vector of optimal simplex multipliers for (LP)} \,\}\ .\end{aligned}$$

■

So, the question of how far $\mathbf{t} = \mathbf{q}+\lambda\mathbf{z}$ is from $\mathrm{zer}(S_{\lambda,A,B})$ essentially becomes: given a primal-dual solution pair $(\mathbf{x}, \pi)$ that is $d_{\max}\|\mathbf{w}\|/\lambda$-balanced and meets $a_{\max}\|\mathbf{w}\|$-

complementary slackness, how far can it be from the set of optimal solution-simplex multiplier pairs ($\mathbf{x}^*$, π^*) of (LP)?

6.3.2. Stability Theorems

To answer the above question, we turn to the existing literature concerning the stability of solutions to linear programs and systems of linear inequalities (Cook *et. al.* 1986, Hoffman 1952, Mangasarian 1981, Mangasarian and Shiau 1987, Robinson 1977, Schrijver 1986, Williams 1963). We will use the results of Cook *et. al.* (1986) and Schrijver (1986), although some of the other cited works could conceivable have been used instead. The key result is as follows:

Theorem 6.3. (Cook *et. al.* 1986, Theorem 5, and Schrijver 1986, Theorem 10.5) Let $\mathbf{Q}$ be a $p \times q$ matrix, and define

$$\Delta(\mathbf{Q}) = \max \{ |(\mathbf{B}^{-1})_{ij}| \mid \mathbf{B} \text{ is a nonsingular square submatrix of } \mathbf{Q} \} .$$

Let $\mathbf{d}$, $\mathbf{r}$, and $\mathbf{r}'$ be conformally-sized vectors such that the linear programs

maximize	$\mathbf{d}^T\mathbf{y}$	maximize	$\mathbf{d}^T\mathbf{y}$
subject to	$\mathbf{Qy} \leq \mathbf{r}$	subject to	$\mathbf{Qy} \leq \mathbf{r}'$

are finite and feasible. Then for each optimal solution $\mathbf{y}'$ of the problem with right hand side $\mathbf{r}'$, there is an optimal solution $\mathbf{y}$ of the problem with right hand side $\mathbf{r}$ such that

$$\| \mathbf{y} - \mathbf{y}' \|_\infty \leq q\Delta(\mathbf{Q}) \| \mathbf{r} - \mathbf{r}' \|_\infty . \blacksquare$$

Note that by setting $\mathbf{d=0}$, we obtain a similar result concerning systems of linear inequalities. The theorem has the pleasing interpretation that the solution set of a linear program or system of linear inequalities is Lipschitz continuous in the right hand side.

The difficulty in using Theorem 6.3 is that we must address the situation in which there is a simultaneous perturbation of both the cost *and* the right hand side. Dualizing the above theorem can only give Lipschitz continuity of the *dual* variables with respect to cost variations. Furthermore, any results we obtain will necessarily be of a more delicate nature than Theorem 6.3, as it is known (Mangasarian and Shiau 1987) that the solution of an LP is *not* in general Lipschitz continuous in the cost coefficients.

From here on, we will restrict the discussion to the special case of *standard primal form* linear programs in which $l_j = 0$ and $u_j = \infty$ for all j. This form is well known to be a completely general linear programming formulation. An analysis is also possible for the general problem (LP), but is much more complicated.

The standard primal form is

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \mathbf{Ax} = \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \quad , \end{array} \tag{SLP}$$

and its dual is

$$\begin{array}{ll} \text{maximize} & \mathbf{b}^T\pi \\ \text{subject to} & \mathbf{A}^T\pi \leq \mathbf{c} \ . \end{array} \tag{DLP}$$

Suppose we have some optimal solutions $(\mathbf{x}', \pi')$ of the perturbed problem

$$\begin{array}{ll} \text{minimize} & (\mathbf{c}')^T\mathbf{x} \\ \text{subject to} & \mathbf{Ax} = \mathbf{b}' \\ & \mathbf{x} \geq \mathbf{0} \end{array} \qquad \text{(SLP')}$$

and its dual

$$\begin{array}{ll} \text{maximize} & (\mathbf{b}')^T\pi \\ \text{subject to} & \mathbf{A}^T\pi \leq \mathbf{c}' \ . \end{array} \qquad \text{(DLP')}$$

We wish to estimate the distance from $(\mathbf{x}', \pi')$ to the set $X^* \times \Pi^*$, where X^* is the set of optima for (SLP), and Π^* that for (DLP). Note that Π^* is also the set of optimal simplex multipliers for (SLP). The following (new) theorem addresses this issue in an important special case.

Theorem 6.4. Let $R_x, R_\pi \geq 1$ be two given numbers. Suppose that there is some optimal solution pair $(\mathbf{x}', \pi')$ of (SLP')-(DLP') such that $\|\mathbf{x}'\|_1 \leq R_x$ and $\|\pi'\|_1 \leq R_\pi$. Then there exists some optimal solution $(\mathbf{x}^*, \pi^*)$ of (SLP)-(DLP) such that

$$\| (\mathbf{x}', \pi') - (\mathbf{x}^*, \pi^*) \|_\infty \ \leq \ (m+n)\Delta(\mathbf{Z}) \, (R_x \| \mathbf{c} - \mathbf{c}' \|_\infty + R_\pi \| \mathbf{b} - \mathbf{b}' \|_\infty) \ ,$$

where $\mathbf{Z}$ is the $(2m+2n+1) \times (m+n)$ matrix

$$\mathbf{Z} \ = \ \begin{bmatrix} -\mathbf{I} & 0 \\ \mathbf{A} & 0 \\ -\mathbf{A} & 0 \\ 0 & \mathbf{A}^T \\ \mathbf{c}^T & -\mathbf{b}^T \end{bmatrix} \ .$$

Proof. $X^* \times \Pi^*$ is exactly the set of solutions to the set of linear inequalities

$$
\begin{array}{rrl}
-\mathbf{I}\mathbf{x} & & \leq \mathbf{0} \\
\mathbf{A}\mathbf{x} & & \leq \mathbf{b} \\
-\mathbf{A}\mathbf{x} & & \leq -\mathbf{b} \\
 & \mathbf{A}^{\mathsf{T}}\pi & \leq \mathbf{c} \\
\mathbf{c}^{\mathsf{T}}\mathbf{x} & - \mathbf{b}^{\mathsf{T}}\pi & \leq 0 \quad ,
\end{array}
$$

which may be written $\mathbf{Z}(\mathbf{x}, \pi) \leq \mathbf{r}$, where $\mathbf{r} = (\mathbf{0}, \mathbf{b}, -\mathbf{b}, \mathbf{c}, 0)$. On the other hand, $(\mathbf{x}', \pi')$ satisfy the slightly different conditions,

$$
\begin{array}{rrl}
-\mathbf{I}\mathbf{x}' & & \leq \mathbf{0} \\
\mathbf{A}\mathbf{x}' & & \leq \mathbf{b}' \\
-\mathbf{A}\mathbf{x}' & & \leq -\mathbf{b}' \\
 & \mathbf{A}^{\mathsf{T}}\pi' & \leq \mathbf{c}' \\
\mathbf{c}'^{\mathsf{T}}\mathbf{x}' & - \mathbf{b}'^{\mathsf{T}}\pi' & \leq 0 \quad .
\end{array}
$$

Rewriting the last inequality as

$$
\mathbf{c}^{\mathsf{T}}\mathbf{x}' - \mathbf{b}^{\mathsf{T}}\pi' \leq (\mathbf{c} - \mathbf{c}')^{\mathsf{T}}\mathbf{x}' - (\mathbf{b} - \mathbf{b}')^{\mathsf{T}}\pi' \quad ,
$$

we conclude that $(\mathbf{x}', \pi')$ is one possible solution in $(\mathbf{x}, \pi)$ of the set of inequalities

$$
\begin{array}{rrl}
-\mathbf{I}\mathbf{x} & & \leq \mathbf{0} \\
\mathbf{A}\mathbf{x} & & \leq \mathbf{b}' \\
-\mathbf{A}\mathbf{x} & & \leq -\mathbf{b}' \\
 & \mathbf{A}^{\mathsf{T}}\pi & \leq \mathbf{c}' \\
\mathbf{c}^{\mathsf{T}}\mathbf{x} & - \mathbf{b}^{\mathsf{T}}\pi & \leq (\mathbf{c} - \mathbf{c}')^{\mathsf{T}}\mathbf{x}' - (\mathbf{b} - \mathbf{b}')^{\mathsf{T}}\pi' \quad ,
\end{array}
$$

which may also be expressed as $\mathbf{Z}(\mathbf{x}, \pi) \leq \mathbf{r}'$, where

$$
\mathbf{r}' = (\mathbf{0}, \mathbf{b}', -\mathbf{b}', \mathbf{c}', (\mathbf{c} - \mathbf{c}')^{\mathsf{T}}\mathbf{x}' - (\mathbf{b} - \mathbf{b}')^{\mathsf{T}}\pi') \, .
$$

By Theorem 6.3, there exists some solution $(\mathbf{x}^*, \pi^*)$ of $\mathbf{Z}(\mathbf{x}, \pi) \leq \mathbf{r}$ such that

$$
\| (\mathbf{x}', \pi') - (\mathbf{x}^*, \pi^*) \|_\infty \leq (m+n)\Delta(\mathbf{Z}) \| \mathbf{r} - \mathbf{r}' \|_\infty \, .
$$

It remains to show that $R_x \| \mathbf{c} - \mathbf{c}' \|_\infty + R_\pi \| \mathbf{b} - \mathbf{b}' \|_\infty \geq \| \mathbf{r} - \mathbf{r}' \|_\infty$. By construction,

$$
\| \mathbf{r} - \mathbf{r}' \|_\infty = \max \{ \| \mathbf{c} - \mathbf{c}' \|_\infty, \| \mathbf{b} - \mathbf{b}' \|_\infty, |(\mathbf{c} - \mathbf{c}')^{\mathsf{T}}\mathbf{x}' - (\mathbf{b} - \mathbf{b}')^{\mathsf{T}}\pi'| \} \, .
$$

Now,

$$\begin{aligned} \left|(\mathbf{c}-\mathbf{c}')^{\mathsf{T}}\mathbf{x}' - (\mathbf{b}-\mathbf{b}')^{\mathsf{T}}\pi'\right| &\leq \| \mathbf{c}-\mathbf{c}' \|_\infty \|\mathbf{x}'\|_1 + \| \mathbf{b}-\mathbf{b}' \|_\infty \|\pi'\|_1 \\ &\leq R_x \| \mathbf{c}-\mathbf{c}' \|_\infty + R_\pi \| \mathbf{b}-\mathbf{b}' \|_\infty \ . \end{aligned}$$

Finally, since $R_x, R_\pi \geq 1$,

$$R_x \| \mathbf{c}-\mathbf{c}' \|_\infty + R_\pi \| \mathbf{b}-\mathbf{b}' \|_\infty \geq \max \{ \| \mathbf{c}-\mathbf{c}' \|_\infty, \| \mathbf{b}-\mathbf{b}' \|_\infty \} \ ,$$

hence $R_x \| \mathbf{c}-\mathbf{c}' \|_\infty + R_\pi \| \mathbf{b}-\mathbf{b}' \|_\infty \geq \| \mathbf{r}-\mathbf{r}' \|_\infty$. ■

At first glance, the conclusion of Theorem 6.4 would seem to contradict the example given in Mangasarian and Shiau (1987), which shows that the optimal solution of an LP cannot in general be Lipschitzian with respect to the cost coefficients, even if the feasible region is bounded. However, there is no contradiction, because the quantity $\Delta(\mathbf{Z})$ depends on $\mathbf{c}$.

Also note that, by substituting more complicated matrices for $\mathbf{Z}$, the theorem could be extended to linear programs with more general upper and lower bound constraints.

6.3.3. Bounding the Primal and Dual Solutions

We now wish to construct bounds R_x and R_π such that $\text{dist}(\mathbf{x}^k, X^*) \leq R_x$ and $\text{dist}(\pi^k, \Pi^*) \leq R_\pi$ for all $k \geq 0$. Given $\{\mathbf{x}^k\}$ and $\{\pi^k\}$, we define $\{\mathbf{p}^k\}$, $\{\mathbf{z}^k\} \subseteq \mathbb{R}^{mn}$ by setting, for all $k \geq 0$,

$$p_{ij}^k = \begin{cases} -\pi_i^k \ , & (i,j) \in Q \\ 0 & (i,j) \notin Q \end{cases}$$

$$z_{ij}^k = \begin{cases} -a_{ij}^k x_j^k + \dfrac{r_i(\mathbf{x}^k)}{q_i} & , \quad (i,j) \in Q \\ 0 & (i,j) \notin Q \end{cases} .$$

Then, as per the discussion of Sections 5.4 and 6.1, $\{\mathbf{x}^k\}$, $\{\mathbf{z}^k\}$, and $\{\mathbf{p}^k\}$ evolve according to

$$\begin{aligned} \mathbf{x}^{k+1} &= \arg\min_{\mathbf{x}} \{f(\mathbf{x}) + \langle \mathbf{p}^k, \mathbf{M}\mathbf{x} \rangle + \tfrac{\lambda}{2} \|\mathbf{M}\mathbf{x} - \mathbf{z}^k\|^2\} \\ \mathbf{z}^{k+1} &= \arg\min_{\mathbf{z}} \{g(\mathbf{z}) - \langle \mathbf{p}^k, \mathbf{z} \rangle + \tfrac{\lambda}{2} \|\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}\|^2\} \qquad (\text{ADMOM}_\lambda) \\ \mathbf{p}^{k+1} &= \mathbf{p}^k + \lambda(\mathbf{M}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}) \end{aligned}$$

and $(\mathbf{p}^k, \mathbf{z}^k) \in B = V^\perp \times C$ for all k.

Let $\mathbf{x}_*$ be the solution of (SLP) with minimum l_2-norm, and let π_* be the solution of (DLP) with minimum l_2-norm. Let $\mathbf{Q}$ be the $m \times m$ diagonal matrix with entries q_i, and $\mathbf{Q}^{1/2}$ be the $m \times m$ diagonal matrix with entries $\sqrt{q_i}$. We then have the technical lemma:

Lemma 6.3. $\operatorname{dist}(\mathbf{0}, \operatorname{zer}(S_{\lambda,A,B})) \leq \delta_* \triangleq \|\mathbf{Q}^{1/2}\pi_*\| + \lambda a_{\max}\|\mathbf{x}_*\|$.

Proof. Define $\mathbf{p}_* \in \mathbb{R}^{mn}$ by

$$p_{*ij} = \begin{cases} -\pi_{*i}^k , & (i,j) \in Q \\ 0 , & (i,j) \notin Q \end{cases} ,$$

and let $\mathbf{t}_* = \mathbf{p}_* + \lambda \mathbf{M}\mathbf{x}_*$. Then $\mathbf{t}_* \in \operatorname{zer}(S_{\lambda,A,B})$ by Lemma 6.2. Also,

$$\begin{aligned} \operatorname{dist}(\mathbf{0}, \operatorname{zer}(S_{\lambda,A,B})) &\leq \|\mathbf{t}_*\| \\ &\leq \|\mathbf{p}_*\| + \lambda\|\mathbf{M}\mathbf{x}_*\| \\ &= \left(\sum_{(i,j) \in Q} \pi_{*i}^2 \right)^{1/2} + \lambda \left(\sum_{j=1}^{n} \sum_{i=1}^{m} a_{ij}^2 x_{*j}^2 \right)^{1/2} \end{aligned}$$

$$= \left(\sum_{i=1}^{m} q_i \pi_{*i}^2 \right)^{1/2} + \lambda \left(\sum_{j=1}^{n} \|\mathbf{a}_j\|^2 x_{*j}^2 \right)^{1/2}$$

$$\leq \|\mathbf{Q}^{1/2}\pi_*\| + \lambda a_{\max}\|\mathbf{x}_*\| \ . \ \blacksquare$$

Now let $a_{\min} = \min_{j=1,\ldots,n} \{\|\mathbf{a}_j\|\}$, and $\mathbf{t}^k = \mathbf{p}^k + \lambda \mathbf{z}^k$ for all $k \geq 0$. We can then prove the following bounds:

Lemma 6.4. $\text{dist}(\mathbf{t}^0, \text{zer}(S_{\lambda,A,B})) \leq \delta_0 \triangleq \delta_* + \|\mathbf{Q}^{1/2}\pi^0\| + \lambda a_{\max}\|\mathbf{x}^0\|$.

Proof. By logic much like that of Lemma 6.3,

$$\|\mathbf{t}^0\| \leq \|\mathbf{p}^0\| + \lambda\|\mathbf{M}\mathbf{x}^0\| \leq \|\mathbf{Q}^{1/2}\pi^0\| + \lambda a_{\max}\|\mathbf{x}^0\| \ .$$

So,

$$\text{dist}(\mathbf{t}^0, \text{zer}(S_{\lambda,A,B})) \leq \|\mathbf{t}^0 - \mathbf{t}_*\| \leq \|\mathbf{t}_*\| + \|\mathbf{t}^0\| \leq \delta_* + \|\mathbf{Q}^{1/2}\pi^0\| + \lambda a_{\max}\|\mathbf{x}^0\| \ . \ \blacksquare$$

Lemma 6.5. The following bounds hold whenever the alternating step method is used on a solvable linear program:

(i) $\|\pi^k\| \leq \delta_0 + \|\mathbf{Q}^{1/2}\pi_*\|$ for all $k \geq 0$

(ii) $\|\mathbf{x}^k\| \leq \dfrac{1}{\lambda a_{\min}}(3\delta_0 + \|\mathbf{Q}^{1/2}\pi_*\|)$ for all $k \geq 1$.

Proof. Because $\{\mathbf{t}^k\}$ evolves according to the proximal point algorithm,

$$\|\mathbf{t}^k - \mathbf{t}_*\| \leq \|\mathbf{t}^0 - \mathbf{t}_*\| \quad \forall\, k \geq 0 \ .$$

Therefore,

$$\|\mathbf{t}^k - \mathbf{t}_*\| \leq \delta_0 = \delta_* + \|\mathbf{Q}^{1/2}\pi^0\| + \lambda a_{\max}\|\mathbf{x}^0\| \quad \forall\, k \geq 0 \ .$$

Applying the nonexpansive operator $J_{\lambda B}$, we have, for all $k \geq 0$,

$$\| \mathbf{p}^k - \mathbf{p}_* \| \leq \delta_0 \quad \Rightarrow \quad \|\mathbf{p}^k\| \leq \delta_0 + \|\mathbf{p}_*\| = \delta_0 + \| \mathbf{Q}^{1/2}\pi_* \| \ .$$

Now $\|\mathbf{p}^k\| = \| \mathbf{Q}^{1/2}\pi^k \| \geq \|\pi^k\|$, so (i) follows.

For $k \geq 1$, we have

$$\begin{aligned} & \mathbf{p}^k = \mathbf{p}^{k-1} + \lambda (\mathbf{M}\mathbf{x}^k - \mathbf{z}^k) \\ \Rightarrow \quad & \mathbf{t}^k = \mathbf{p}^k + \lambda \mathbf{z}^k = \mathbf{p}^{k-1} + \lambda \mathbf{M}\mathbf{x}^k \ . \end{aligned}$$

Thus,

$$\begin{aligned} & \|\mathbf{t}^k\| \leq \delta_0 + \|\mathbf{t}_*\| = \delta_0 + \delta_* \leq 2\delta_0 \\ \Rightarrow \quad & \| \mathbf{p}^{k-1} + \lambda \mathbf{M}\mathbf{x}^k \| \leq 2\delta_0 \\ \Rightarrow \quad & \lambda \| \mathbf{M}\mathbf{x}^k \| \leq 2\delta_0 + \|\mathbf{p}^{k-1}\| \leq 2\delta_0 + \delta_0 + \| \mathbf{Q}^{1/2}\pi_* \| \\ \Rightarrow \quad & \lambda \| \mathbf{M}\mathbf{x}^k \| \leq 3\delta_0 + \| \mathbf{Q}^{1/2}\pi_* \| \\ \Rightarrow \quad & \lambda a_{\min} \|\mathbf{x}^k\| \leq 3\delta_0 + \| \mathbf{Q}^{1/2}\pi_* \| \\ \Rightarrow \quad & \|\mathbf{x}^k\| \leq \frac{1}{\lambda a_{\min}} (3\delta_0 + \| \mathbf{Q}^{1/2}\pi_* \|) \ . \ \blacksquare \end{aligned}$$

6.3.4. Convergence Rate Theorem

We are now ready to approach the main convergence rate result. Let

$$R_x = \max \left\{ 1, \ \frac{\sqrt{n}}{\lambda a_{\min}} \left(3\delta_0 + \|\mathbf{Q}^{1/2}\pi_*\| \right) \right\}$$

$$R_\pi = \max \left\{ 1, \ \sqrt{m} \left(\delta_0 + \|\mathbf{Q}^{1/2}\pi_*\| \right) \right\} \ .$$

Then R_x and R_π are bounds on $\|\mathbf{x}^k\|_1$ for all $k \geq 1$, and $\|\pi^k\|_1$ for all $k \geq 0$, respectively. Furthermore, let

$$\gamma = (m+n)\Delta(\mathbf{Z})\,[a_{\max}R_x + \frac{d_{\max}}{\lambda}R_\pi]$$

$$\alpha = \left(\sqrt{mn} + \lambda\left[\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right]^{1/2}\right)\gamma$$

$$\tau = \frac{\alpha}{\sqrt{\alpha^2+1}} < 1 \ .$$

Theorem 6.5. When applying the alternating step method to a solvable standard-form linear program (SLP), the following inequalities hold

(i) $\quad \mathrm{dist}(\pi^k, \Pi^*) \le \tau^k \delta_0 \quad \forall\, k \ge 0$

(ii) $\quad \mathrm{dist}_\infty(\mathbf{x}^k, X^*) \le \tau^{k-1}(\gamma\delta_0) \quad \forall\, k \ge 1$,

where $\mathrm{dist}_\infty(\mathbf{x}^k, X^*)$ denotes $\min\{\|\mathbf{x}^k - \mathbf{x}\|_\infty \mid \mathbf{x} \in X^*\}$. In addition,

(iii) $\quad \mathrm{dist}(\mathbf{t}^k, \mathrm{zer}(S_{\lambda,A,B})) \le \tau\,\mathrm{dist}(\mathbf{t}^{k-1}, \mathrm{zer}(S_{\lambda,A,B})) \quad \forall\, k \ge 1$

(iv) $\quad \mathrm{dist}(\mathbf{t}^k, \mathrm{zer}(S_{\lambda,A,B})) \le \tau^k \delta_0 \quad \forall\, k \ge 0$,

where $\{\mathbf{t}^k\} = \{\mathbf{p}^k + \lambda\mathbf{z}^k\}$ is defined as above.

Proof. Note that from the construction of the alternating direction method of multipliers (see Section 3.5.6), we have $(\mathbf{q}^k, -\mathbf{M}\mathbf{x}^k) \in A$ for all $k \ge 1$, where

$$\mathbf{q}^k = \mathbf{p}^{k-1} + \lambda(\mathbf{M}\mathbf{x}^k - \mathbf{z}^{k-1}) \ .$$

Now, the proximal point iteration $\mathbf{t}^{k+1} = (I + S_{\lambda,A,B})^{-1}(\mathbf{t}^k)$ implies that

$$\mathbf{t}^{k-1} - \mathbf{t}^k \in S_{\lambda,A,B}(\mathbf{t}^k) \quad \forall\, k \ge 1 \ .$$

Let $\mathbf{t} = \mathbf{t}^k$, and

$$\begin{aligned} \mathbf{w} &= \mathbf{t}^{k-1} - \mathbf{t}^k \\ &= (\mathbf{p}^{k-1} + \lambda\mathbf{z}^{k-1}) - (\mathbf{p}^k + \lambda\mathbf{z}^k) \end{aligned}$$

$$
\begin{aligned}
&= (\mathbf{p}^{k-1} - \mathbf{p}^k) + \lambda(\mathbf{z}^{k-1} - \mathbf{z}^k) \\
&= \lambda(\mathbf{z}^k - \mathbf{M}\mathbf{x}^k) + \lambda(\mathbf{z}^{k-1} - \mathbf{z}^k) \\
&= \lambda(\mathbf{z}^{k-1} - \mathbf{M}\mathbf{x}^k) \\
&= \mathbf{p}^{k-1} - \mathbf{q}^k \quad .
\end{aligned}
$$

Then we have $(\mathbf{t}, \mathbf{w}) \in S_{\lambda,A,B}$. Let

$$
\begin{aligned}
\mathbf{p} &= \mathbf{p}^{k-1} & \mathbf{s} &= -\mathbf{M}\mathbf{x}^k \\
\mathbf{z} &= \mathbf{z}^{k-1} & \mathbf{x} &= \mathbf{x}^k \\
\mathbf{q} &= \mathbf{q}^k & \pi &= \pi^{k-1} \quad .
\end{aligned}
$$

Then we have $(\mathbf{p}, \mathbf{z}) \in B$, $(\mathbf{q}, \mathbf{s}) \in A$,

$$
\mathbf{q} + \lambda\mathbf{s} = \mathbf{p}^{k-1} + \lambda(\mathbf{M}\mathbf{x}^k - \mathbf{z}^{k-1}) - \lambda\mathbf{M}\mathbf{x}^k = \mathbf{p}^{k-1} - \lambda\mathbf{z}^{k-1} = \mathbf{p} - \lambda\mathbf{z} \ ,
$$

and

$$
\mathbf{q} + \lambda\mathbf{z} = \mathbf{p}^{k-1} + \lambda(\mathbf{M}\mathbf{x}^k - \mathbf{z}^{k-1}) + \lambda\mathbf{z}^{k-1} = \mathbf{p}^{k-1} + \lambda\mathbf{M}\mathbf{x}^k = \mathbf{t}^k = \mathbf{t} \quad .
$$

So, $\mathbf{p}$, $\mathbf{q}$, $\mathbf{s}$, $\mathbf{z}$, $\mathbf{x}$ and π meet the conditions of Lemma 6.1. Therefore, we deduce that

$\mathbf{x} = \mathbf{x}^k$	is $d_{\max}\|\mathbf{t}^{k-1} - \mathbf{t}^k\|/\lambda$-balanced
$(\mathbf{x}, \pi) = (\mathbf{x}^k, \pi^{k-1})$	obeys $a_{\max}\|\mathbf{t}^{k-1} - \mathbf{t}^k\|$-complementary slackness.

Thus, $(\mathbf{x}^k, \pi^{k-1})$ are jointly optimal for some pair of linear programs

$$
\begin{aligned}
&\text{minimize} && (\mathbf{c}')^T\mathbf{x} \\
&\text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{b}' \\
& && \mathbf{x} \geq \mathbf{0}
\end{aligned}
\qquad \text{(SLP')}
$$

$$
\begin{aligned}
&\text{maximize} && (\mathbf{b}')^T\pi \\
&\text{subject to} && \mathbf{A}^T\pi \leq \mathbf{c}' \ ,
\end{aligned}
\qquad \text{(DLP')}
$$

where $\|\mathbf{c} - \mathbf{c}'\|_\infty \le a_{\max}\|\mathbf{t}^{k-1} - \mathbf{t}^k\|$ and $\|\mathbf{b} - \mathbf{b}'\|_\infty \le d_{\max}\|\mathbf{t}^{k-1} - \mathbf{t}^k\|/\lambda$. Theorem 6.4 then gives, by the construction of R_x and R_π, that there exists $(\mathbf{x}^*, \pi^*) \in X^* \times \Pi^*$ such that

$$\max\{\|\mathbf{x}^k - \mathbf{x}^*\|_\infty, \|\pi^{k-1} - \pi^*\|_\infty\} \le \left((m+n)\Delta(\mathbf{Z})\left[a_{\max}R_x + \frac{d_{\max}}{\lambda}R_\pi\right]\right)\|\mathbf{t}^{k-1} - \mathbf{t}^k\|$$

$$= \gamma\|\mathbf{t}^{k-1} - \mathbf{t}^k\| \ .$$

Defining $\mathbf{p}^*$ by

$$p_{ij}^* = \begin{cases} -\pi_i^*, & (i,j) \in Q \\ 0, & (i,j) \notin Q \end{cases} ,$$

we have that $\mathbf{t}^* = \mathbf{p}^* + \lambda\mathbf{M}\mathbf{x}^* \in \text{zer}(S_{\lambda,A,B})$. Also,

$$\begin{aligned}
\|\mathbf{t}^k - \mathbf{t}^*\| &\le \|\mathbf{p}^{k-1} - \mathbf{p}^*\| + \lambda\|\mathbf{M}\mathbf{x}^k - \mathbf{M}\mathbf{x}^*\| \\
&\le \|\mathbf{p}^{k-1} - \mathbf{p}^*\| + \lambda\|\mathbf{M}(\mathbf{x}^k - \mathbf{x}^*)\| \\
&\le \sqrt{mn}\,\|\mathbf{p}^{k-1} - \mathbf{p}^*\|_\infty + \lambda\left[\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right]^{1/2}\|\mathbf{x}^k - \mathbf{x}^*\|_\infty \\
&= \sqrt{mn}\,\|\pi^{k-1} - \pi^*\|_\infty + \lambda\left[\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right]^{1/2}\|\mathbf{x}^k - \mathbf{x}^*\|_\infty \\
&\le \sqrt{mn}\gamma\|\mathbf{t}^{k-1} - \mathbf{t}^k\| + \lambda\left[\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2\right]^{1/2}\gamma\|\mathbf{t}^{k-1} - \mathbf{t}^k\| \\
&= \alpha\|\mathbf{t}^{k-1} - \mathbf{t}^k\| \ .
\end{aligned}$$

Thus,

$$\text{dist}(\mathbf{t}^k, \text{zer}(S_{\lambda,A,B})) \le \alpha\|\mathbf{t}^{k-1} - \mathbf{t}^k\| \ .$$

As $\{\mathbf{t}^k\}$ is generated by the proximal point algorithm with stepsize 1 for all k, we have from Luque (1984a) that

$$\text{dist}^2(\mathbf{t}^k, \text{zer}(S_{\lambda,A,B})) \leq \text{dist}^2(\mathbf{t}^{k-1}, \text{zer}(S_{\lambda,A,B})) - \|\mathbf{t}^{k-1} - \mathbf{t}^k\|^2 ,$$

hence

$$\text{dist}^2(\mathbf{t}^k, \text{zer}(S_{\lambda,A,B})) \leq \text{dist}^2(\mathbf{t}^{k-1}, \text{zer}(S_{\lambda,A,B})) - \frac{1}{\alpha^2}\text{dist}^2(\mathbf{t}^k, \text{zer}(S_{\lambda,A,B})) .$$

Simplifying and taking the square root, we obtain

$$\text{dist}(\mathbf{t}^k, \text{zer}(S_{\lambda,A,B})) \leq \left(\frac{\alpha}{\sqrt{\alpha^2+1}}\right)\text{dist}(\mathbf{t}^{k-1}, \text{zer}(S_{\lambda,A,B})) \quad \forall\ k \geq 1 ,$$

which is (iii). By induction and the Lemma 6.4, we immediately obtain (iv).

We now demonstrate (i) and (ii). For all $k \geq 0$, there exists some $\mathbf{t}^{*k} \in \text{zer}(S_{\lambda,A,B})$ such that $\|\mathbf{t}^k - \mathbf{t}^{*k}\| \leq \tau^k\delta_0$. Applying the nonexpansive operator $J_{\lambda B}$ to $\mathbf{t}^k$ and $\mathbf{t}^{*k}$, one obtains $\|\mathbf{p}^k - \mathbf{p}^{*k}\| \leq \tau^k\delta_0$, where $\mathbf{p}^{*k}$ and $\mathbf{z}^{*k}$ are such that

$$\mathbf{t}^{*k} = \mathbf{p}^{*k} + \lambda\mathbf{z}^{*k}$$
$$(\mathbf{p}^{*k}, \mathbf{z}^{*k}) \in B$$
$$(\mathbf{p}^{*k}, -\mathbf{z}^{*k}) \in A .$$

It follows that $\|\pi^k - \pi^{*k}\| \leq \tau^k\delta_0$, where $\pi^{*k} \in \Pi^*$ is defined by $\pi_i{}^{*k} = -p_{ij}{}^{*k}$ for all $(i, j) \in Q$. This fact implies (i). Finally, for $k \geq 1$, we again have from Luque (1984a) that

$$\begin{aligned}\|\mathbf{t}^{k-1} - \mathbf{t}^k\|^2 &\leq \text{dist}^2(\mathbf{t}^{k-1}, \text{zer}(S_{\lambda,A,B})) - \text{dist}^2(\mathbf{t}^k, \text{zer}(S_{\lambda,A,B})) \\ &\leq \text{dist}^2(\mathbf{t}^{k-1}, \text{zer}(S_{\lambda,A,B})) \\ &\leq (\tau^{k-1}\delta_0)^2 .\end{aligned}$$

Taking the square root and using that there exists some $\mathbf{x}^{*k} \in X^*$ such that

$$\|\mathbf{x}^k - \mathbf{x}^{*k}\|_\infty \leq \gamma\|\mathbf{t}^{k-1} - \mathbf{t}^k\| ,$$

we get (ii). ■

Thus, we have shown that the alternating step method converges linearly from the start when applied to linear programs in standard form. Note, however, that the bound τ given for the convergence factor can be *extremely* close to 1. Thus, Theorem 6.5 is too weak a result to provide a proof of, say, polynomial complexity. Even so, linear convergence is certainly a desirable property.

6.4. Complexity of the Iteration and Theoretical Efficiency of Parallel Implementations

Even though our linear convergence rate result does not imply overall polynomiality, it is still appropriate to consider the complexity of *each iteration* of the alternating step method, and how this complexity may be improved by parallel implementation.

Both the alternating step method and generalized alternating step method iterations are highly parallelizable. Fundamentally, this potential for parallelism exists because the calculations of the x_j^{k+1}, $j=1,...,n$, are independent of one another, and can therefore be performed simultaneously. Likewise, the calculations of the π_i^{k+1}, $i = 1,...,m$, are also independent, and can be done in parallel for all m. In the generalized method, the additional calculation $\mathbf{y}^{k+1} = (1 - \rho_k)\mathbf{y}^k + \rho_k\mathbf{x}^{k+1}$ breaks down into n independent calculations $y_j^{k+1} = (1 - \rho_k)y_j^k + \rho_k x_j^{k+1}$, which, again, may be performed at the same time.

In studying parallel implementations of the alternating step iteration, we will employ the basic terminology and computational model of Bertsekas and Tsitsiklis (1989), Chapter 1:

Definition 6.3. For a given problem instance, the *speedup* attained by a parallel implementation of an algorithm $\mathcal{A}$ using p processors is

$$S_p = \frac{T_p}{T_1} \quad ,$$

where T_p is the time taken by the parallel implementation of $\mathcal{A}$, and T_1 is the time taken by a serial, one-processor implementation of $\mathcal{A}$. The *efficiency* of the p-processor implementation is

$$\mathcal{E}_p = \frac{S_p}{p} \quad .$$

Note that we are measuring speedup and efficiency relative to a serial implementation of *the same algorithm*, rather than the theoretically best serial algorithm. Intuitively, the efficiency $\mathcal{E}_p$ measures the fraction of the time that an "average" processor in the parallel implementation spends doing useful work.

We use that standard O, Ω, Θ notation for rates of growth (see *e.g.* Papadimitriou and Steiglitz 1982, Chapter 8), and assume that the basic arithmetic operations of addition, subtraction, comparison, multiplication, and division are implemented in $\Theta(1)$ time.

We now consider parallel implementations of the iterations (ASLP) and (GASLP). For general linear programs, we distinguish between two basic representations of the constraint matrix **A**: the *dense* representation, in which **A** is stored in computer memory as an $m \times n$ array, including all zero elements, and a *sparse* representation. For the purposes of discussion, assume that the sparse representation consists of a linked list for each row of **A**, and a linked list for each column. Pointers to the heads of these lists are

stored in m- and n-element arrays, respectively, so the heads of all lists are accessible in $\Theta(1)$ time. Each list contain an entry only for each non-zero entry a_{ij} in the associated row or column, and the entry includes the values of i, j, and a_{ij}.

Let J denote the total number of non-zero entries in **A**.

Definition 6.2. The *valence* $v(j)$ of variable x_j is the number of nonzero components in column j of **A** (this notion is dual to that of the degree of a constraint). Let

$$v_{\max} = \max \{v(j) \mid j=1,...,n\}$$

$$d_{\max} = \max \{d(i) \mid i=1,...,m\} \quad .$$

The definition of $d_{\max}$ is identical to that of Lemma 6.1.

Proposition 6.3. Using the dense representation of **A**, one iteration of algorithm (ASLP) or (GASLP) requires $\Theta(mn)$ time in a serial environment. One serially-implemented iteration of either algorithm takes $\Theta(J)$ time using the sparse data representation.

Proof. These statements should be clear from the description of the iterations. To expand, consider first the dense case of (ASLP) first. Computation of each of the $r_i(\mathbf{x}^k)$, $i = 1,...,m$, requires $\Theta(n)$ time, for a total of $\Theta(mn)$ time. The calculation of each of the n quantities

$$x_j^{k+1} = P_{V(j)}\left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\left[\sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{x}^k)}{q_i}\right] - \frac{\bar{c}_j(\pi^k)}{\lambda}\right)\right]$$

requires $\Theta(m)$ time, again for a total of $\Theta(mn)$. One must then recalculate the surpluses $r_i(\mathbf{x}^{k+1})$, again requiring $\Theta(mn)$ time. Note, however, that these values can be saved for

the next iteration. Once the surpluses are known, computing the π^{k+1} requires $\Theta(m)$ time. The total time is thus $\Theta(mn) + \Theta(mn) + \Theta(m) = \Theta(mn)$.

In the sparse case of (ASLP), calculation of each surplus $r_i(\mathbf{x}^k)$ takes only $\Theta(d(i))$ time, for a total of $\Theta\left(\sum_{i=1}^{m} d(i)\right) = \Theta(J)$. Similarly, the x_j^{k+1} computations now require $\Theta\left(\sum_{j=1}^{n} v(j)\right) = \Theta(J)$, and the total time is $\Theta(J) + \Theta(J) + \Theta(m) = \Theta(J)$.

For (GASLP), the analysis is virtually identical, except for an additional $\Theta(n)$ time to compute the y_j^{k+1}, which does not worsen either the dense or sparse complexity. ■

6.4.1. The Bipartite and Overlap Implementations

One of the most natural parallel implementations of the alternating step method involves assigning one processor to each variable and one processor to each constraint, for a total of $n+m$ processors. Each variable processor stores, for a given j, the values of x_j^k, the values of $r_i(\mathbf{x}^k)$, and π_i^k for all i such that $a_{ij} \neq 0$, and, in the case of the generalized method, the value of y_j^k. The processor for constraint i stores π_i^k, $r_i(\mathbf{x}^k)$, and also the x_j^k for all j such that $a_{ij} \neq 0$. The iteration proceeds as follows: each of the n variable processors computes x_j^{k+1} for a particular j, and then communicates the result to the $v(j)$ constraint processors for which $a_{ij} \neq 0$. Then, each constraint processor computes $r_i(\mathbf{x}^{k+1})$ and π_i^{k+1} for a particular i, and sends both results to all $d(i)$ variable processors for which $a_{ij} \neq 0$. The communication paths needed by this implementation form a bipartite graph, hence we call it the *bipartite implementation.* A path between constraint processor i and variable processor j need be present only if $a_{ij} \neq 0$, so the communication graph is essentially the sparsity graph of **A**. We do not necessarily require that there be a direct physical link between variable each variable processor and constraint processor

that need to communicate. Instead, we use the term "bipartite implementation" for any arrangement in which the processors are divided into two sets, one processor from the first set being assigned to each variable, and one processor from the second set being assigned to each constraint, regardless of the physical communication topology. Figure 18 depicts the bipartite implementation.

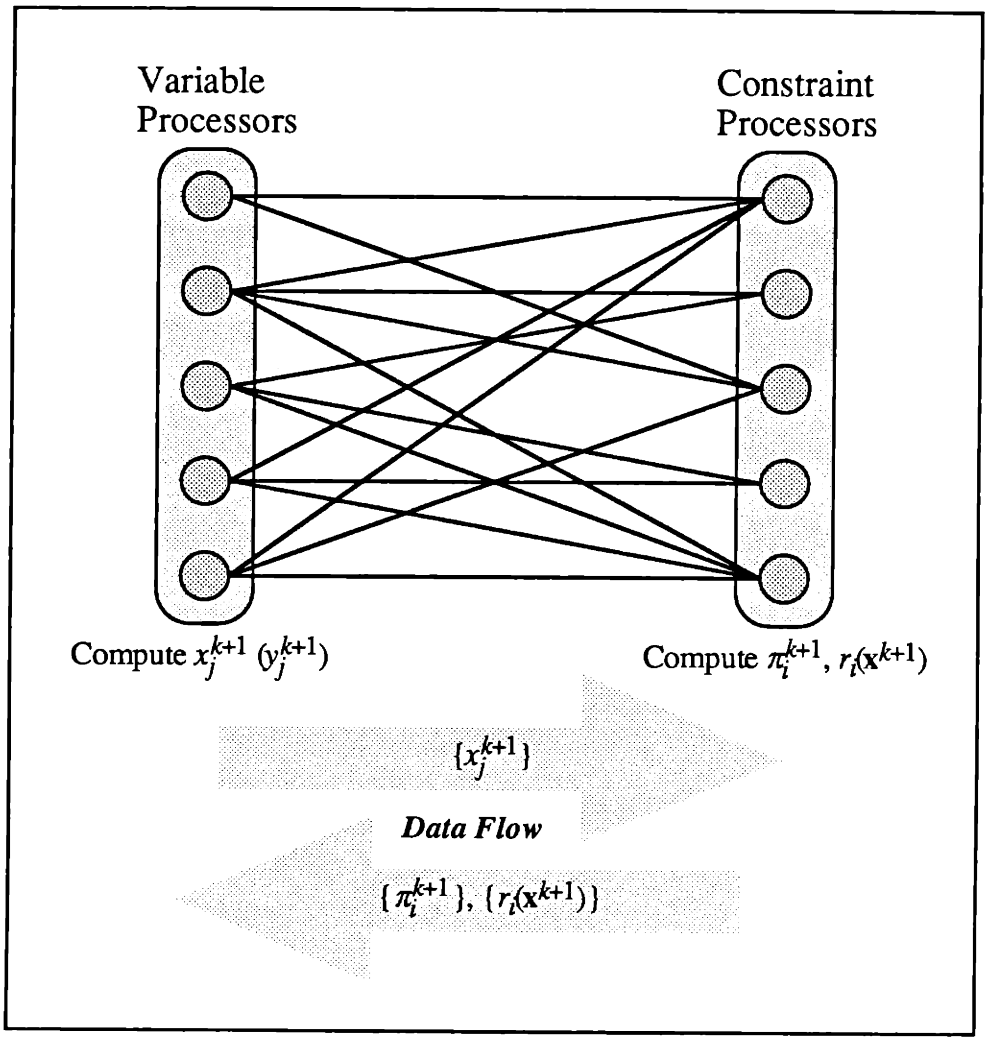


Figure 18. The bipartite implementation of the alternating step method. In the generalized alternating step method, the variable processors also store and update the y_j^k variables.

A careful examination of the alternating step iteration reveals that it is not actually necessary for the variable processor j to store *both* π_i^k and $r_i(\mathbf{x}^k)$ for all i such that $a_{ij} \neq 0$, nor is it necessary for constraint processor i to broadcast both π_i^k and $r_i(\mathbf{x}^k)$. Instead, the constraint processors need only transmit the quantities

$$\alpha_i^{k+1} = \frac{r_i(\mathbf{x}^{k+1})}{q_i} + \frac{\pi_i^{k+1}}{\lambda} \quad .$$

Variable processor j may then compute

$$\begin{aligned} x_j^{k+1} &= P_{V(j)}\left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\sum_{i=1}^{m} a_{ij}\alpha_i^k - \frac{c_j}{\lambda} \right) \right] \\ &= P_{V(j)}\left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{x}^k)}{q_i} + \frac{1}{\lambda}\left[\sum_{i=1}^{m} a_{ij}\pi_i^k - c_j \right] \right) \right] \\ &= P_{V(j)}\left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\left[\sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{x}^k)}{q_i} \right] - \frac{\bar{c}_j(\pi^k)}{\lambda} \right) \right] \quad , \end{aligned}$$

and need only store the α_i^k, for i such that $a_{ij} \neq 0$, as opposed to both π_i^k and $r_i(\mathbf{x}^k)$.

Proposition 6.4. Assuming $O(1)$ communication delays, the dense bipartite implementation of (ASLP) — or (GASLP) — can execute one iteration in $O(n+m)$ time, with a speedup of

$$S_{n+m} = \frac{\Theta(nm)}{O(n+m)} = \Omega\left(\frac{nm}{n+m}\right) \quad ,$$

while the efficiency of the implementation is

$$\mathcal{E}_{n+m} = \frac{\Theta(nm)}{(n+m) \cdot O(n+m)} = \Omega\left(\frac{nm}{(n+m)^2}\right) \quad .$$

With a sparse data representation, the bipartite implementation executes an iteration in $O(v_{\max}+d_{\max})$ time, with a speedup of

$$S_{n+m} = \frac{\Theta(J)}{O(v_{\max} + d_{\max})} = \Omega\left(\frac{J}{v_{\max} + d_{\max}}\right) ,$$

and an efficiency of

$$\mathcal{E}_{n+m} = \frac{\Theta(J)}{(n+m) \cdot O(v_{\max} + d_{\max})} = \Omega\left(\frac{J}{(n+m)(v_{\max} + d_{\max})}\right) .$$

Each iteration of either the dense of sparse iteration requires that $\Theta(J)$ messages be sent between processors. If, instead of $O(1)$ delays, one supposes that the $n+m$ processors in the bipartite implementation are connected by a linear array, ring, binary balanced tree, mesh, or hypercube communication network, then the iteration can still be implemented in $O(n+m)$ time, including communications.

Proof. Consider the dense case first. All n updates of the x_j^{k+1} (and y_j^{k+1}, if necessary) may be done simultaneously, collectively requiring $O(m)$ time. Likewise, all m computations of the surpluses can be done in parallel, consuming $O(n)$ time. Subsequently, the computations of the π_i^{k+1}, $i = 1,...,m$, may be done in parallel in $O(1)$ time. Thus, the time per iteration is $O(m) + O(n) + O(1) = O(n+m)$. The speedup and efficiency results then follow by direct calculation. In the sparse case, we can again do all the x_j^{k+1} updates in parallel, the most time-consuming one requiring $O(v_{\max})$ time. Similarly, the calculation of the surpluses and the update of the dual variables now takes $O(d_{\max}) + O(1) = O(d_{\max})$ time, for a total of $O(v_{\max}+d_{\max})$. Again, the speedup and efficiency statements follow immediately. The message complexity of both forms of the iteration follows because, for each of the J pairs (i, j) such that $a_{ij} \neq 0$, constraint processor i and variable processor j must exchange either two or three values per iteration: x_j^{k+1} and α_i^{k+1}, or x_j^{k+1}, π_i^{k+1}, and $r_i(\mathbf{x}^{k+1})$. (Note that there is never any need to communicate the y_j^{k+1}.)

Now consider imposing the bipartite implementation on a set of $n+m$ processors with a linear array, ring, binary balanced tree, mesh, or hypercube physical communication

topology. Two communication functions are required for each iteration: sending the values of the x_j^{k+1} to the constraint processors, and sending the α_i^{k+1} to the variable processors. In the worst case, both functions can be accomplished by multinode broadcast operations, which require $O(n+m)$ time, or less, for all the topologies mentioned (see Bertsekas and Tsitsiklis 1989, Section 1.3.4). Since the computations also require at most $O(n+m)$ time, the total time is $O(n+m)$, including communication. ■

For specific sparsity structures of **A** and specific physical communication topologies between the $n+m$ processors, it should be possible to reduce the communication time needed by the bipartite approach to less than that needed for a multinode broadcast. The multinode broadcast time merely provides a simple upper bound, showing, at least in the dense case, that communication need not be a theoretical bottleneck for the implementation.

The efficiency of the bipartite implementation can be improved by using the same physical processors for both the constraint and variable calculations, reducing the number of processors required to $\max\{n, m\}$. We call this technique the *overlap* implementation. The following proposition may be confirmed by direct computation:

Proposition 6.5. In both the dense and sparse cases, the message complexities, speedups, and iteration time complexities (both excluding communication, and including it for linear, ring, balanced binary tree, and hypercube topologies) of the overlap implementation are identical to those of the bipartite implementation. However, its efficiency is

$$\mathcal{E}_{\max\{n,m\}} = \Omega\left(\frac{nm}{\max\{n, m\} \cdot (n+m)}\right) = \Omega\left(\frac{\min\{n, m\}}{n+m}\right)$$

in the dense case, and

$$\mathcal{E}_{\max\{n,m\}} = \Omega\left(\frac{J}{\max\{n, m\} \cdot (v_{\max} + d_{\max})}\right)$$

in the sparse case.

Intuitively, the following result says that in a sequence of problems in which the number of constraints and the number of variables grow at roughly the same rate, the efficiencies of the dense bipartite and overlap implementations have the desirable property of not approaching zero in the limit:

Proposition 6.6. If $m = \Theta(n)$, the efficiencies of the dense bipartite and dense overlap implementations of the (generalized) alternating step method stay bounded away from zero.

Proof. In the bipartite case,

$$\mathcal{E}_{n+m} = \Omega\left(\frac{nm}{(n+m)^2}\right) = \Omega\left(\frac{n^2}{n^2}\right) = \Omega(1) \quad ,$$

while in the overlap case,

$$\mathcal{E}_{\max\{n,m\}} = \Omega\left(\frac{\min\{n, m\}}{n+m}\right) = \Omega\left(\frac{n}{n}\right) = \Omega(1) \quad . \quad \blacksquare$$

6.4.2. An Implementation for Network Flow Problems

In situations where the maximum valence or degree remains bounded as problems increase in size, the bipartite and overlap implementations of the alternating step method may not be efficient. Consider a sequence of completely dense network problems with N

nodes and $A = N^2$ arcs. Suppose that we assign one processor to each variable (arc), and one to each constraint (node), for a total of $n + m = N^2 + N$ processors. Then, the N^2 arc processors will only be active for $\Theta(1)$ time per iteration, but each iteration takes $\Theta(N)$ time, so the implementation's efficiency will approach zero as N approaches infinity. The same conclusion may be deduced from the formula for $\mathcal{E}_{n+m}$ above, and the situation for the overlap implementation is similar.

An alternate implementation for network problems with N nodes and A arcs is as follows: one processor is assigned to each node, and the communication network between processors is assumed to be the same as the problem network, where all arcs are treated as bidirectional communication links. Let $\{\mathcal{L}(v)\}_{v=1,\ldots,N}$ be a partition of the network's arc set $\mathcal{L}$ such that, for every node v, all arcs in $\mathcal{L}(v)$ are incident to v. Let $I(v)$ denote the set of all arcs incident to v. At the k^{th} iteration, the processor for node v computes, using the data from neighboring nodes in the network, the flow x_{vw}^{k+1} or x_{wv}^{k+1} for each arc (v, w) or (w, v) in $\mathcal{L}(v)$. For each such arc, it then informs node w of the new value of the flow. It then receives from neighboring processors updated values for the flows on all arcs in $I(v) \setminus \mathcal{L}(v)$. Then, it calculates $r_v(\mathbf{x}^{k+1})$ and π_v^{k+1}, and transmits these quantities to all neighboring nodes w such that $(v, w) \in I(v) \setminus \mathcal{L}(v)$ or $(w, v) \in I(v) \setminus \mathcal{L}(v)$. We call this arrangement the *dense network implementation*. For the generalized alternating step method, the processor for node v is also charged with calculating y_{vw}^{k+1} and y_{wv}^{k+1} for all $(v, w), (w, v) \in \mathcal{L}(v)$.

Proposition 6.7. The dense network implementation achieves a speedup of $\Omega(A/d_{\max})$ over the sparse serial implementation, and an efficiency of $\Omega(\bar{d}/d_{\max})$, where $\bar{d} = 2A/N$ is the average degree of a node in the problem network. Its iteration time complexity is $O(d_{\max})$ and its iteration message complexity is $O(A)$.

Proof. Observe that $J = 2A$. The computation of the surpluses and update of the dual variables takes $O(d_{\max}) + O(1) = O(d_{\max})$ time, just as in the sparse bipartite implementation. For the processor at node v to update the flow variables for all arcs in $\mathcal{L}(v)$ takes $O(|\mathcal{L}(v)|)$ time. Since $|\mathcal{L}(v)| \leq d(v) \leq d_{\max}$ for all v, and these updates occur simultaneously for all nodes v, the primal updates now require $O(d_{\max})$ time as well. Thus the total time per iteration is $O(d_{\max})$. A serial implementation requires $O(A)$ time per iteration, so the speedup is

$$\mathcal{S}_N = \Omega\left(\frac{J}{d_{\max}}\right) = \Omega\left(\frac{A}{d_{\max}}\right) \quad .$$

The efficiency then works out to

$$\mathcal{E}_N = \Omega\left(\frac{A}{N d_{\max}}\right) = \Omega\left(\frac{\bar{d}}{d_{\max}}\right) \quad .$$

The message complexity should be clear from the description of the algorithm: for each arc (v, w), two messages are sent per iteration. One contains the flow on the arc, and the other the price and surplus of the incident node that does not "own" the arc. ■

When applied to completely dense network problems, the dense network implementation attains $\Theta(1)$ efficiency. There are still some sequences of problems for which the efficiency may tend to zero, but in networks with special structure, better speedup and efficiency bounds than $\Theta(A/d_{\max})$ and $\Theta(\bar{d}/d_{\max})$ may be possible by choosing the partition $\{\mathcal{L}(v)\}_{v \in \mathcal{N}}$ cleverly.

The dense network implementation requires a communication topology conforming to that of the problem network. In practice, it may not be possible to embed such a topology, particularly a dense one, in a realistic physical processor configuration, such as a

hypercube. However, in an N-processor hypercube, one could still imagine assigning one processor to each problem node, while using a total exchange communication procedure (see Bertsekas and Tsitsiklis 1989, Section 1.3.4) to transmit the arc flows x_j^{k+1}, and a multinode broadcast to transmit the α_i^{k+1}. These communication procedures can be implemented in $O(N)$ and $O(N/\log N)$ time, respectively, so that the total time per iteration is $O(N)$. For dense problem networks, this complexity is essentially the same as the complexity discounting communication time, $O(d_{\max})$.

6.4.3. Low-Level Parallelism

It is possible to implement the alternating step method using more that $n+m$ processors. Consider the calculation of a single surplus $r_i(\mathbf{x}^{k+1})$:

$$r_i(x_j^{k+1}) \quad = \quad b_i \quad - \sum_{j:\, a_{ij} \neq 0} a_{ij} x_j^{k+1} \quad .$$

The principal task here is to calculation the inner product of $\mathbf{A}_i$ and $\mathbf{x}^{k+1}$. This task may be accomplished in $\Theta(\log d(i))$ time using $d(i)$ processors, as suggested in Bertsekas and Tsitsiklis (1989), Chapter 1. It is thus possible to compute all the quantities $r_i(\mathbf{x}^{k+1})$, π_i^{k+1}, and $\alpha_i^{k+1} = r_i(\mathbf{x}^{k+1})/q_i + \pi_i^{k+1}/\lambda$ in $\Theta(\log d(i)) + \Theta(1) = \Theta(\log d(i))$ time using $d(i)$ processors. An example directed acyclic graph (DAG) representation of this computation is shown in Figure 19. Note that we take advantage of $\mathbf{A}_i$ being fixed throughout each run of the algorithm, and do not allocate processors to indices j for which $a_{ij} = 0$. See Bertsekas and Tsitsiklis (1989), Chapter 1, for an explanation of the DAG representation of algorithms.

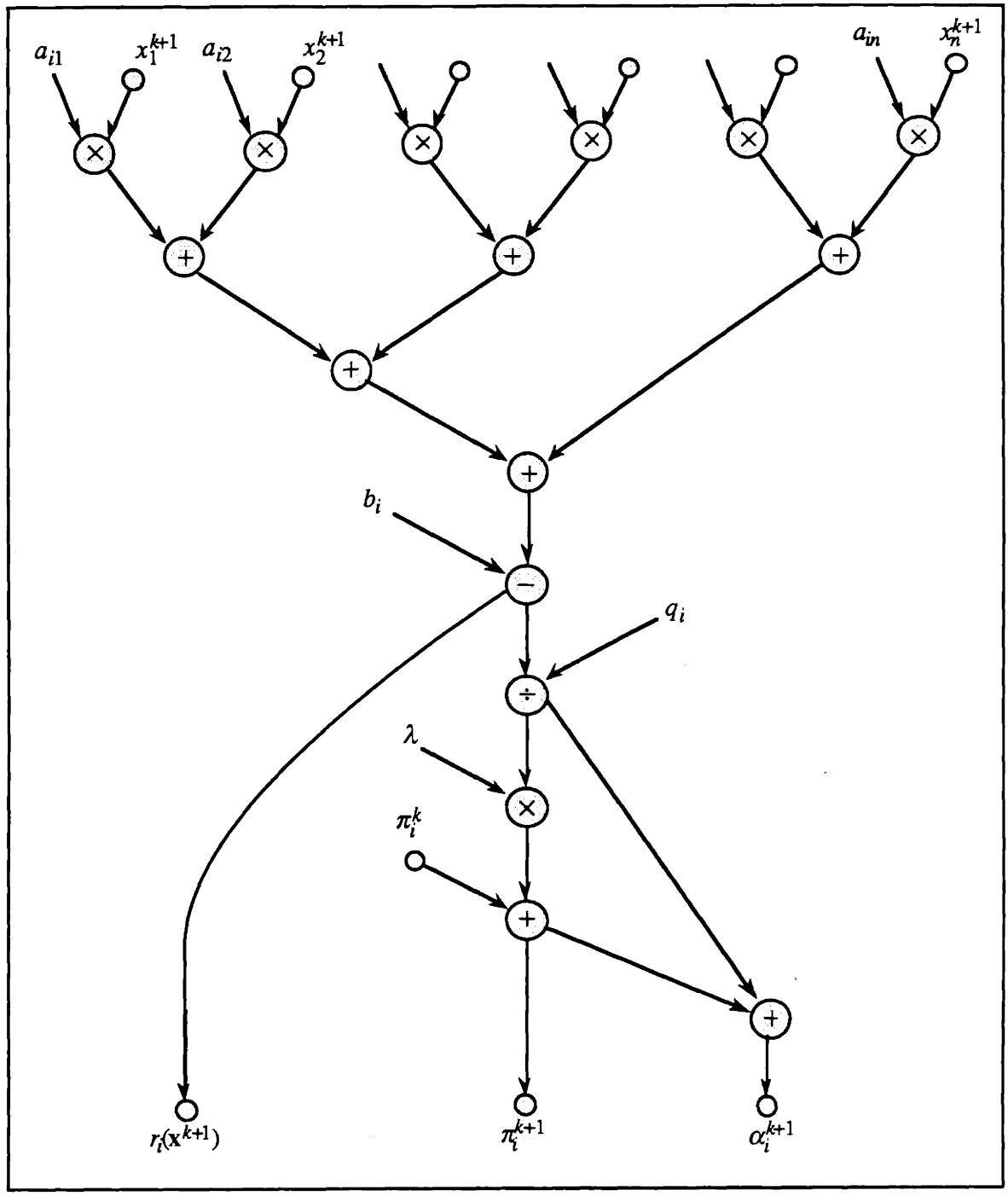


Figure 19. DAG representation of a possible computation of $r_i(\mathbf{x}^{k+1})$, π_i^{k+1}, and α_i^{k+1} in $\Theta(\log d(i))$ time using $d(i)$ processors. The diagram may also be viewed as a systolic array of $\Theta(d(i))$ processors.

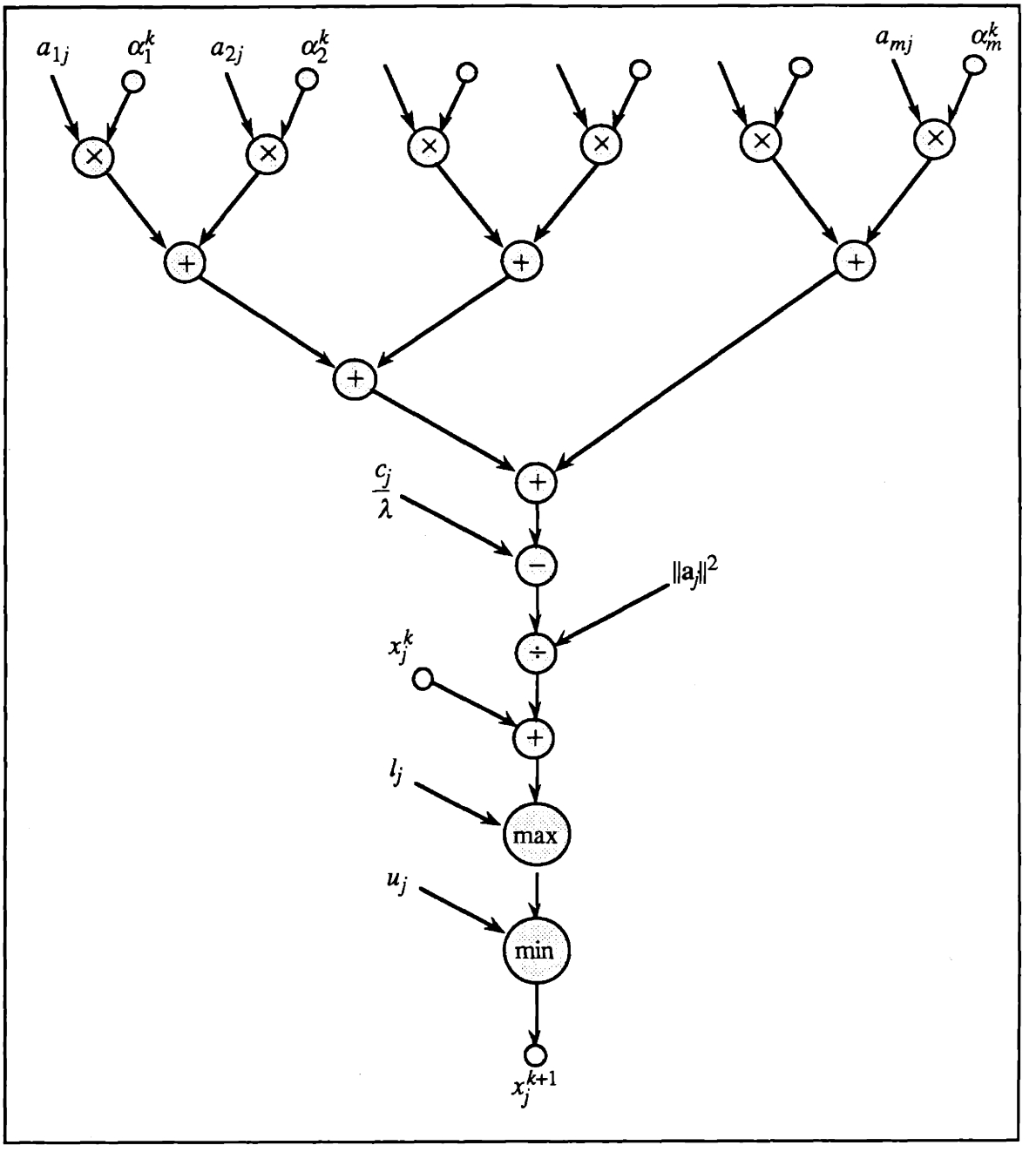


Figure 20. Schematic DAG representation of a possible computation of x_j^{k+1} with $\nu(j)$ processors in $\Theta(\nu(j))$ time.

Combining in parallel m processor configurations of the type shown in Figure 19, one can compute all m surpluses $r_i(\mathbf{x}^{k+1})$, dual variables π_i^{k+1}, and auxiliary variables α_i^{k+1} in $\Theta(d_{\max})$ time using $\sum_{i=1}^{m} d(i) = J$ processors. Unfortunately, the number of messages needed climbs to $\Theta(\sum_{i=1}^{m} d(i) \log d(i))$, or roughly $O(J \log d_{\max})$.

We now consider the update of primal x_j^k variables, which may be written, as in Section 6.4.1,

$$x_j^{k+1} = P_{V(j)}\left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\sum_{i=1}^{m} a_{ij}\alpha_i^k - \frac{c_j}{\lambda}\right)\right] .$$

The principal work here is forming the inner product of $\mathbf{a}_j$ with the vector $\alpha^k = (\alpha_1^k, \ldots, \alpha_m^k)$. Figure 20 shows a DAG illustrating how $v(j)$ processors can perform this calculation in $\Theta(\log v(j))$ time. Much as for Figure 19, we use that the matrix $\mathbf{A}$ is fixed, and do not allocate processors for indices i such that $a_{ij} = 0$. For the generalized alternating step method, an additional $\Theta(1)$ time would be needed to update y_j^{k+1}.

Combining in parallel n such configurations like that of Figure 20, it is possible to compute $\mathbf{x}^{k+1}$ in $O(\log v_{\max})$ time using $\sum_{j=1}^{n} v(j) = J$ processors. Combining this result with that for the computation of the surpluses and dual variables, we have shown

Proposition 6.8. Using J processors, and with $O(1)$ communication delays, it is possible to execute the alternating step or generalized alternating step method iteration in

$$\Theta(\log d_{\max} + \log v_{\max}) = \Theta(\log(d_{\max} v_{\max}))$$

time, with $\Theta(J \log(d_{\max} v_{\max}))$ messages being sent. The speedup of such an implementation over the sparse serial method is $\Theta(J / \log(d_{\max} v_{\max}))$, and its efficiency is only $\Theta(1 / \log(d_{\max} v_{\max}))$.

In reality, there are a myriad of different conceivable parallel implementations of the alternating step method. Let $\mathbf{Q}$ be the $m \times m$ diagonal matrix with entries q_i, and let $\mathbf{D}$ be the $n \times n$ diagonal matrix with entries $\|\mathbf{a}_j\|^{-2}$. If we let V be the "box-shaped" set $V(1) \times \ldots \times V(n) \subseteq \mathbb{R}^n$, and let s_i^k denote $r_i(\mathbf{x}^k)/q_i$, then we can write the alternating step iteration as

$$\begin{aligned}
\mathbf{x}^{k+1} &= P_V(\mathbf{x}^k + \mathbf{D}(\mathbf{A}^\top \alpha^k - \mathbf{c})) \\
\mathbf{s}^{k+1} &= \mathbf{Q}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}) \\
\pi^{k+1} &= \pi^k + \lambda \mathbf{s}^{k+1} \\
\alpha^{k+1} &= \mathbf{s}^{k+1} + \frac{1}{\lambda}\pi^{k+1} \quad .
\end{aligned}$$

Thus, most parallel techniques for doing vector addition and matrix-vector multiplication should be applicable to the method. We have given just a few possible implementations. In a specific computing environment, communication/concurrency trade-offs might dictate other approaches (see Bertsekas and Tsitsiklis (1989), Section 1.3.5).

Chapter 7

Computational Experiments

This chapter reviews actual computational experience with the alternating step method for linear programming. As seen in Chapter 6, the alternating step method is highly amenable to parallel implementation, the main question being whether it converges rapidly enough to become, with parallelism, a competitive method. For example, if its convergence rate were to be as slow in reality as the bound proved in Section 6.3, then it would take so many iterations to approach an optimal solution that, even with massive parallelism, the method would not likely be of much use.

Section 7.1 introduces the notion of *spiralling* convergence of the alternating step method, and shows how this problem may be related to a generic difficulty with the convergence rate of Douglas-Rachford splitting schemes. Section 7.2 gives computational results for assignment problems, for which the convergence of the alternating step method does not seem to be much affected by spiralling. We demonstrate how to appropriately choose the various parameters of the algorithm so as to obtain satisfactory convergence on reasonably sparse assignment problems. Section 7.3 introduces the *alternating direction transportation algorithm*, an alternative application of the alternating direction method of multipliers to assignment and transportation problems. We show that this algorithm performs very similarly to the alternating step method. In Section 7.4, we study the performance of the alternating step method on transportation problems. We find that even for problems with relatively small levels of supply and demand at each node, the algorithm

behaves in a radically different manner than it does for assignment problems, converging very slowly.

Finally, Section 7.5 discusses some actual implementations of the alternating step method on real parallel computer systems. The main difficulty is in finding a computer architecture and algorithm implementation that permit massive parallelism without imposing communication bottlenecks. We will see that on the one hand, the *Alliant FX/8* computer system imposes virtually no communication overhead, but can provide only moderate parallelism, whereas on the other hand, the *Connection Machine 2* computer system promotes massive parallelism, but does not seem to permit a commensurately high rate of communication.

7.1. Spiralling

In analyzing the convergence of the alternating step method, one must use two measures of closeness to optimality: a measure of primal feasibility and a measure of complementary slackness. A measure of primal feasibility is fairly straightforward to define: we set

$$\bar{r}(k) = \| \mathbf{r}(\mathbf{x}^k) \|_\infty = \| \mathbf{b} - \mathbf{A}\mathbf{x}^k \|_\infty = \max_{i=1,\ldots,m} \{ | r_i(\mathbf{x}^k) | \} \quad .$$

That is, $\bar{r}(k)$ is the largest amount by which any of the constraints $\mathbf{A}_i^T \mathbf{x}^k = b_i$ is violated.

For $\mathbf{x}^k$ to be an optimal solution, it needs not only to be feasible, but also to satisfy complementary slackness in conjunction with some vector of simplex multipliers. Accordingly, we define $s_j(k)$ to be the following function of $\mathbf{x}^k$ and π^k:

$$s_j(k) = \begin{cases} \bar{c}_j(\pi^k) , & l_j < x_j^k < u_j \\ \max \{ \bar{c}_j(\pi^k), 0 \} , & x_j^k = u_j \\ \min \{ \bar{c}_j(\pi^k), 0 \} , & x_j^k = l_j \end{cases} \quad .$$

We then let

$$\varepsilon(k) = \max_{j=1,\ldots,n} \{ |s_j(k)| \} \quad .$$

Proposition 7.1. The pair $(\mathbf{x}^k, \pi^k)$ always obeys $\varepsilon(k)$-complementary slackness, that is

$$x_j^k > l_j \quad \Rightarrow \quad \bar{c}_j(\pi^k) \leq \varepsilon(k)$$
$$x_j^k < u_j \quad \Rightarrow \quad \bar{c}_j(\pi^k) \geq -\varepsilon(k) \quad .$$

Furthermore, $(\mathbf{x}^k, \pi^k)$ is an optimal primal solution/simplex multiplier pair if and only if $\bar{r}(k) = \varepsilon(k) = 0$.

Proof. The first statement follows by the construction of the $s_j(k)$ and of $\varepsilon(k)$. Also, $(\mathbf{x}^k, \pi^k)$ is an optimal pair if and only if it obeys complementary slackness and $\mathbf{A}\mathbf{x}^k = \mathbf{b}$. The former is true if and only if $\varepsilon(k) = 0$, and the latter if and only if $\bar{r}(k) = 0$. ■

We do not necessarily propose to run the alternating step method until $\bar{r}(k)$ and $\varepsilon(k)$ are both zero to machine precision; we may instead terminate when both $\bar{r}(k)$ and $\varepsilon(k)$ are very small numbers. The main point is that $\bar{r}(k)$ and $\varepsilon(k)$ are useful measures of how far $\mathbf{x}^k$ and π^k are from optimality. Figure 21 shows a typical plot of $\bar{r}(k)$ and $\varepsilon(k)$ versus k for a small transportation problem.

The pattern revealed in Figure 21 is fairly typical for small transportation problems. At first, both $\bar{r}(k)$ and $\varepsilon(k)$ decrease roughly monotonically with k. Shortly, however, the algorithm settles into a pattern in which the graphs of $\bar{r}(k)$ and $\varepsilon(k)$ resemble "bouncing balls" of gradually decreasing amplitude and opposite phase. Since *both* $\bar{r}(k)$ and $\varepsilon(k)$ must be small for one to be sure of being near a solution, the result is slow convergence.

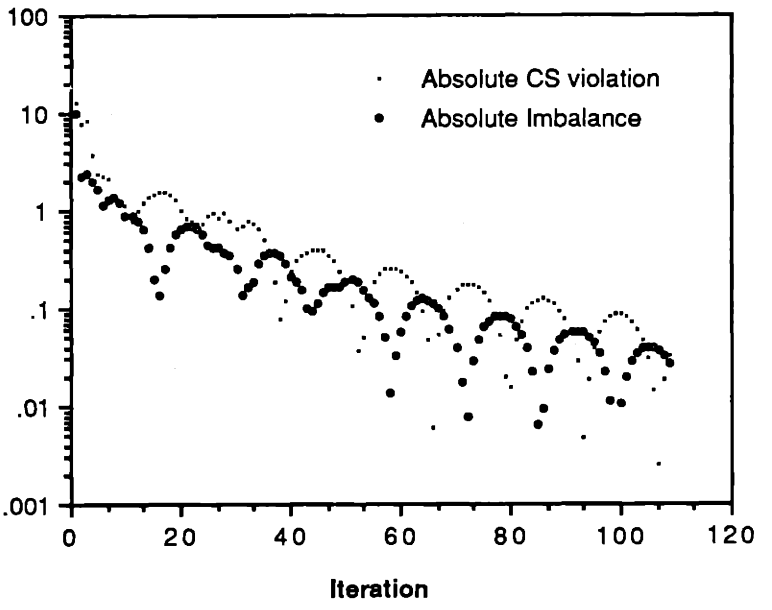


Figure 21. Plot of the maximum absolute surplus $\bar{r}(k)$ and the maximum absolute complementary slackness violation $\varepsilon(k)$ versus the iteration number k, on a logarithmic scale. Data are from an actual run of the alternating step method on a small transportation problem.

Some light may be shed on this phenomenon by plotting *signed* versions of $\bar{r}(k)$ and $\varepsilon(k)$.

Let

$$i_{\max}(k) = \arg\max_{i=1,\ldots,m} \left\{ \left| r_i(\mathbf{x}^k) \right| \right\}$$
$$r_{\max}(k) = r_{[i_{\max}(k)]}(\mathbf{x}^k) \quad ,$$

so that $\bar{r}(k) = |r_{\max}(k)|$. Also, let

$$j_{\max}(k) = \arg\max_{j=1,\ldots,n} \left\{ \left| s_j(k) \right| \right\}$$
$$\sigma(k) = s_{[j_{\max}(k)]}(k) \quad ,$$

so that $\varepsilon(k) = |\sigma(k)|$. Figure 22 shows a typical plot of $r_{\max}(k)$ versus $\sigma(k)$ for large k.

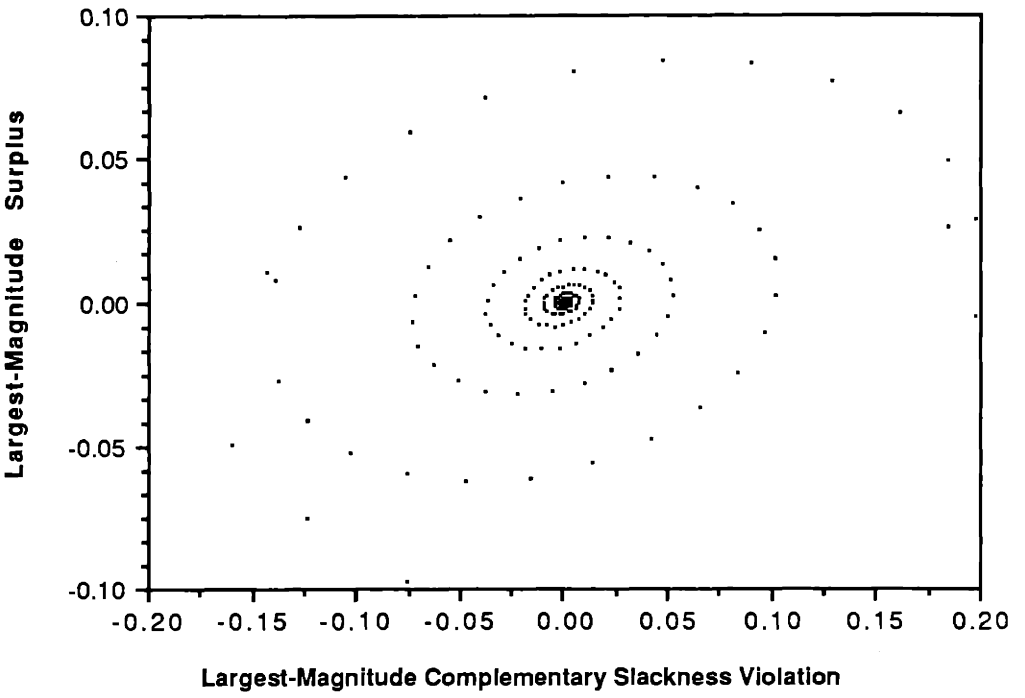


Figure 22: Data from a run of the alternating step method on a small transportation problem, depicting the largest-magnitude surplus $r_{\max}(k)$ versus the the largest-magnitude complementary slackness violation $\sigma(k)$ for large k.

Figure 22 reveals that $\{(r_{\max}(k), \sigma(k))\}$ forms a gradual spiral sequence leading towards (0, 0). Despite extensive investigation, the precise conditions that promote this kind of spiralling are not yet understood. However, this sort of gradual convergence does appear to be an intrinsic problem with Douglas-Rachford-based methods, as we will now illustrate.

Consider applying the Douglas-Rachford splitting algorithm to the operators $A = N_U$ and $B = N_W$ of Proposition 6.1. That is, $A = U \times U^{\perp}$ and $B = W \times W^{\perp}$, where

$$W = \{(x_1, x_2) \mid x_2 = 0\} = \{(x_1, 0) \mid x_1 \in \mathbb{R}\} \subseteq \mathbb{R}^2$$
$$U = \{(x_1, x_2) \mid x_2 = \mu x_1\} = \{(x, \mu x) \mid x \in \mathbb{R}\} \subseteq \mathbb{R}^2$$
$$W^{\perp} = \{(x_1, x_2) \mid x_1 = 0\} = \{(0, x_2) \mid x_2 \in \mathbb{R}\} \subseteq \mathbb{R}^2$$
$$U^{\perp} = \{(x_1, x_2) \mid x_2 = -\tfrac{1}{\mu} x_1\} = \{(z, -\tfrac{1}{\mu} z) \mid z \in \mathbb{R}\} \subseteq \mathbb{R}^2 \ .$$

The only zero of $A+B = N_U + N_W$ is the sole point in $U \cap W$, namely (0, 0). With the stepsize λ set to 1, the Douglas-Rachford iteration then proceeds as follows: given any $(x^k, 0) \in W$ and $(0, b^k) \in W^{\perp}$, find $(v_1{}^{k+1}, v_2{}^{k+1}) \in U$ and $(a_1{}^{k+1}, a_2{}^{k+1}) \in U^{\perp}$ such that

$$(v_1{}^{k+1} + a_1{}^{k+1}, v_2{}^{k+1} + a_2{}^{k+1}) = (x^k, 0) - (0, b^k) = (x^k, -b^k) \ .$$

Then, find $(x^{k+1}, 0) \in W$ and $(0, b^{k+1}) \in W^{\perp}$ such that

$$(x^{k+1}, 0) + (0, b^{k+1}) = (v_1{}^{k+1}, v_2{}^{k+1}) + (0, b^k) \ ,$$

that is, $(x^{k+1}, b^{k+1}) = (v_1{}^{k+1}, v_2{}^{k+1} + b^k)$.

Now, the task of finding $(v_1{}^{k+1}, v_2{}^{k+1})$ is simply that of projecting $(x^k, -b^k)$ onto the line $v_2 = \mu v_1$, so

$$(v_1^{k+1}, v_2^{k+1}) = \left(\frac{x^k - \mu b^k}{1 + \mu^2}\right)(1, \mu) \ .$$

Thus,

$$\begin{aligned}(x^{k+1}, b^{k+1}) &= (v_1^{k+1}, v_2^{k+1} + b^k) \\ &= \left(\frac{x^k - \mu b^k}{1 + \mu^2}, \ \frac{\mu x^k - \mu^2 b^k}{1 + \mu^2} + \frac{b^k + \mu^2 b^k}{1 + \mu^2}\right) \\ &= \left(\frac{1}{1 + \mu^2}\right)(x^k - \mu b^k, \mu x^k + b^k)\end{aligned}$$

$$= \frac{1}{1+\mu^2}\begin{bmatrix} 1 & -\mu \\ \mu & 1 \end{bmatrix}(x^k, b^k) \quad .$$

Thus (x^k, b^k) develops according to the linear recursion $(x^{k+1}, b^{k+1}) = \mathbf{W}(x^k, b^k)$, where $\mathbf{W}$ is the 2×2 matrix

$$\mathbf{W} = \frac{1}{1+\mu^2}\begin{bmatrix} 1 & -\mu \\ \mu & 1 \end{bmatrix} \quad .$$

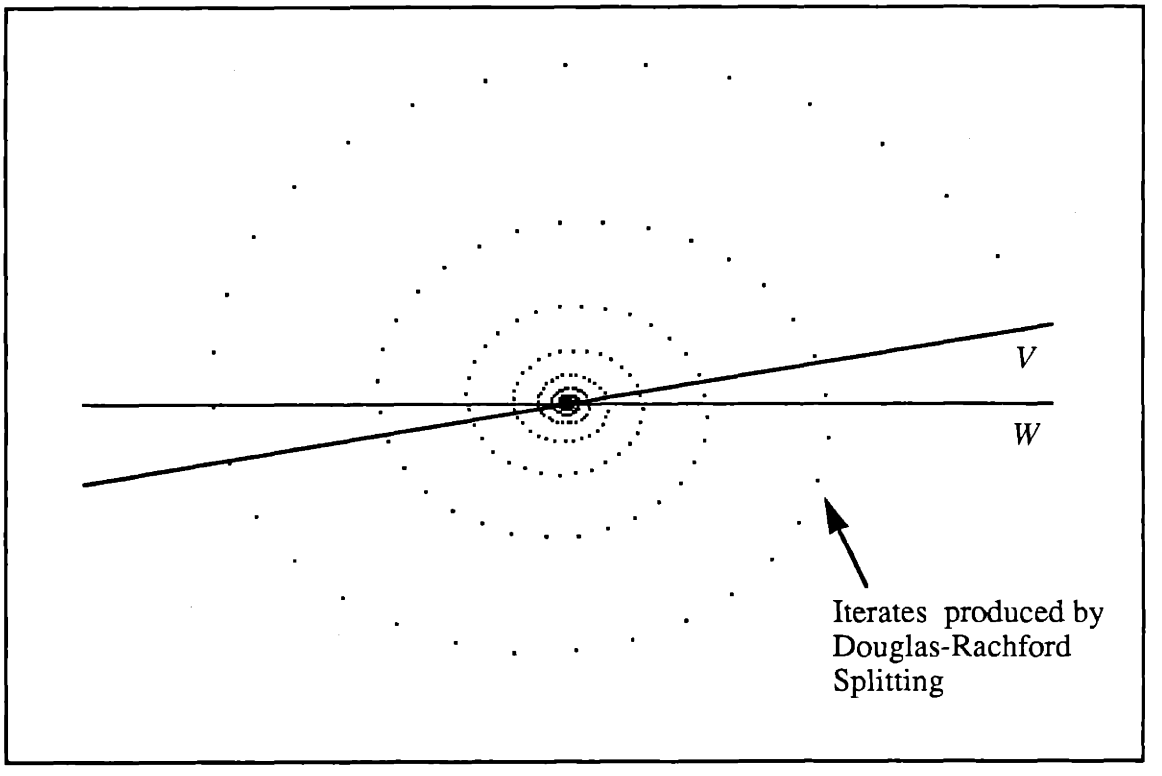


Figure 23. Prototypical slow convergence example for Douglas-Rachford splitting.

By direct calculation, the eigenvalues of $\mathbf{W}$ are $(1 \pm \mu i)/(1 + \mu^2)$, where i denotes the square root of -1. Both eigenvalues have magnitude $(1 + \mu^2)^{-1/2}$. By choosing μ to be a very small positive number, one can make $(1 + \mu^2)^{-1/2}$ arbitrarily close to 1, and thus obtain arbitrarily slow convergence. Figure 23 shows a plot of $(x^0, b^0),\ldots,(x^{350}, b^{350})$ for

$\mu = 0.2$. The resemblance to Figure 22 is striking. One therefore surmises that similar convergence mechanisms are probably at work in both cases.

As an aside, we note that for $\mu > 0$, the Douglas-Rachford splitting scheme fails to converge finitely for $A = N_U$ and $B = N_W$. Proposition 4.11 gives that the method is equivalent to applying the proximal point algorithm to the partial inverse operator $(\partial\delta_W)_U$. We thus have an example of infinite convergence of the method of partial inverses as applied to a polyhedral convex function (namely δ_W) which is much simpler than that of Durier and Michelot (1986).

7.2. Computational Experiments with Assignment Problems

Curiously, the slow spiralling convergence phenomenon of Figure 22 does not seem to afflict the alternating step method for linear programming when the method is applied to *assignment* problems. We will show that the alternating step method can converge to high precision in a reasonable number of iterations for such problems. Our formulation of the assignment problem is as follows:

Let

$\mathcal{V}$ be a finite set of source nodes

$\mathcal{W}$ be the set of sink nodes, of the same cardinality as $\mathcal{V}$

$\mathcal{L}$ be a subset of $\mathcal{V} \times \mathcal{W}$.

The assignment problem is then

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(v,w) \in \mathcal{L}} c_{vw} x_{vw} \\
\text{subject to} \quad & \sum_{w:\,(v,w) \in \mathcal{L}} x_{vw} = 1 \qquad \forall\ v \in \mathcal{V} \\
& \sum_{v:\,(v,w) \in \mathcal{L}} x_{vw} = 1 \qquad \forall\ w \in \mathcal{W} \\
& 0 \le x_{vw} \le 1 \qquad\qquad \forall\ (v,w) \in \mathcal{L} \quad .
\end{aligned}
\tag{AP}
$$

The upper bounds $x_{vw} \le 1$ for all $(v, w) \in \mathcal{L}$ are redundant. However, they were found to dramatically speed up convergence of the alternating step method, and so were retained in all computational tests.

In our initial tests, we applied the alternating step method for linear programming, without modification, to the formulation (AP). Two questions immediately arise: how should one choose the $\{q_i\}$, and how should one set the parameter λ? We address the former question first.

7.2.1. Choosing the $\{q_i\}$

Each q_i, $i = 1,...,m$, must be an integer between the degree $d(i)$ of the corresponding constraint and the total number of primal variables n, inclusive. Table 1 compares four strategies for picking the q_i in assignment problems. Each strategy was applied to a set of seven sparse assignment problems with between 200 and 1024 source nodes and integer arc costs uniformly distributed over [1, 100]. All of the problems were created using NETGEN (Klingman *et. al.* 1974). The table shows the average number of iterations, over the seven problems, to reduce $\bar{r}(k)$ and $\varepsilon(k)$ to 0.001 or less, starting from $\mathbf{x}^0 = \mathbf{0}$ and $\pi^0 = \mathbf{0}$.

Strategy	Average Number of Iterations
q_i = degree $d(v)$ or $d(w)$ of associated node	894
q_i = twice degree of associated node	1311
q_i = maximum degree $d_{\max}$ of problem network	1576
q_i = number of arcs	More than 5000

Table 1. Evaluation of strategies for choosing the q_i.

From the table, we see that the setting q_i to the minimum possible value, the degree of the associated constraint (or, equivalently, node), is clearly the most efficient strategy. This result accords with intuition, as the minimum possible choice for q_i permits the largest possible adjustments to be made in the primal step

$$x_j^{k+1} = P_{V(j)}\left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\left[\sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{x}^k)}{q_i}\right] - \frac{\bar{c}_j(\pi^k)}{\lambda}\right)\right]$$

and the dual step

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda}{q_i} r_i(\mathbf{x}^{k+1}) \ .$$

7.2.2. Choosing λ

Another major issue in the alternating step method is that of choosing the parameter λ. A large value of λ allows large adjustments in the dual step

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda}{q_i} r_i(\mathbf{x}^{k+1}) \ ,$$

but permits only small adjustments proportional to the reduced cost in the primal step

$$x_j^{k+1} = P_{V(j)}\left[x_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\left[\sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{x}^k)}{q_i}\right] - \frac{\bar{c}_j(\pi^k)}{\lambda}\right)\right] \quad .$$

Conversely, a small value of λ permits large primal adjustments, but only small steps for the dual vector π^k. Thus, one needs to choose λ neither too large nor too small.

We have found that a simple, effective strategy for choosing λ is to set $\lambda = \theta c_{\max}$, where

$$c_{\max} = \|\mathbf{c}\|_\infty = \max_{j=1,\ldots,n} \{|c_j|\}$$

is the maximum absolute value of the arc costs, and θ is some positive constant. Typically, values of θ between 1/10 and 1/3 seem to yield the best results. In the example shown in Table 2, the alternating step method was applied to a collection of six NETGEN-generated assignment problems with between 100 and 125 supply nodes, and arc cost ranges varying from [1, 100] to [1, 3200]. For each value of θ, we ran the alternating step method with $\lambda = \theta c_{\max}$ on all six problems, and recorded the average number of iterations to reduce $\bar{r}(k)$ and $\varepsilon(k)$ to 0.001. Figure 24 presents a graph of these data.

In the rest of this chapter, we will set q_i equal to the degree $d(i)$ of the corresponding constraint for all i, and choose λ in the range $[(0.1)c_{\max}, (0.3)c_{\max}]$. As seen from Figure 24, the performance of the alternating step method begins to degrade rapidly when λ is outside this range.

θ	Average Number of Iterations
0.025	758
0.050	438
0.075	342
0.100	302
0.150	257
0.200	268
0.300	285
0.500	338
1.000	508

Table 2. Evaluation of $\lambda = \theta c_{\max}$ for for various values of θ.

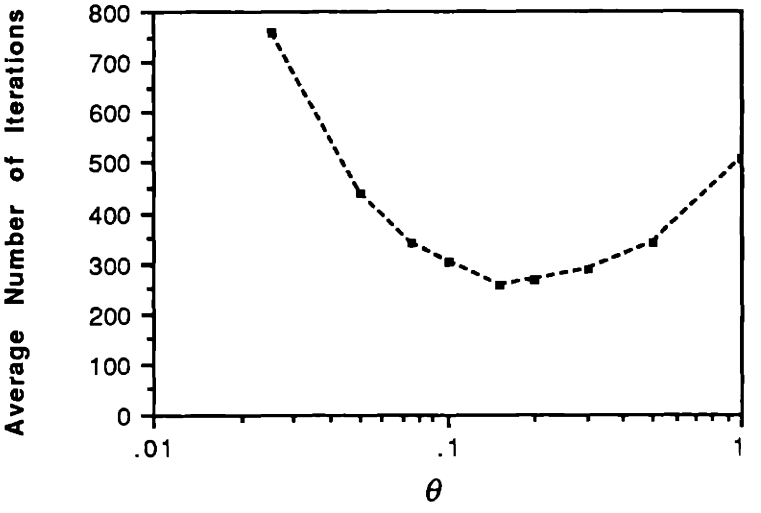


Figure 24. Plot of data from Table 2. Reasonable choices of θ appear to be in the range [0.1, 0.3].

7.2.3. Setting the Relaxation Factor

If we consider the *generalized* alternating step method for linear programming,

$$x_j^{k+1} = P_{V(j)}\left[y_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\left[\sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{y}^k)}{q_i}\right] - \frac{\bar{c}_j(\pi^k)}{\lambda}\right)\right] \qquad j=1,\ldots,n$$

$$y_j^{k+1} = (1-\rho_k) y_j^k + \rho_k x_j^{k+1} \qquad j=1,\ldots,n$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda \rho_k}{q_i} r_i(\mathbf{x}^{k+1}) \qquad i=1,\ldots,m \quad ,$$

another obvious question arises: how should one choose the under/over-relaxation factors $\{\rho_k\}_{k=0}^{\infty}$? We will discuss only strategies of the form $\rho_k = \rho$ for all k, where $\rho \in (0, 2)$ is fixed. Investigation of strategies where ρ_k varies with k must be left for future research, although a few preliminary test did not reveal any consistent benefit from such approaches.

Table 3 gives the results of trying different values of ρ on a pool of 22 different NETGEN assignment problems. For each value of ρ, we recorded the average number of iterations to reduce the maximum absolute surplus $\bar{r}(k)$ and the maximum absolute complementary slackness violation $\varepsilon(k)$ to 0.001 or less. The problems had between 8 and 2048 total nodes, 100 and 5120 arcs, and cost ranges varying from [1, 100] to [1, 3200]. We used $\lambda = (0.15)c_{\max}$ throughout.

Figure 25 illustrates the results. Evidently, values of ρ between 1 and 2 are preferable, with the optimum somewhere in the neighborhood of 1.5. Performance degrades rapidly for values of ρ that less than 1. Accordingly, all subsequent runs in this chapter were done with $\rho = 1$ or $\rho = 1.5$. Note that running the generalized method with $\rho = 1$ is equivalent to using the regular alternating step method.

ρ	Average Number of Iterations
0.50	1040
0.75	703
1.00	556
1.25	485
1.50	469
1.75	563

Table 3. Evaluation of relaxation strategies of the form $\rho_k = \rho$ for all k on a pool of 22 dissimilar assignment problems. In each case, we set $\lambda = (0.15)c_{\max}$.

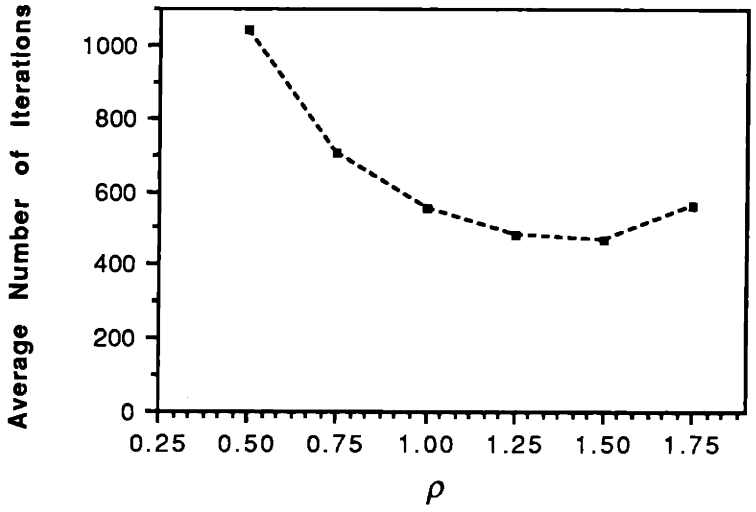


Figure 25. Plot of data from Table 3. It appears that ρ should be at least 1.

7.2.4. An Initialization Procedure: The Twin Lambda Heuristic

In all the runs shown so far, we used the arbitrary starting point $\pi^0 = \mathbf{0}$ and $\mathbf{x}^0 = \mathbf{0}$ (or $\mathbf{y}^0 = \mathbf{0}$ in the case of the generalized method). We also tried a number of more careful initialization procedures in the hopes of improving the run time of the method. These procedures included heuristics that have proven useful in other dual assignment algorithms, such as AUCTION (Bertsekas 1979b, 1985, 1988, Bertsekas and Eckstein 1986, 1987). For instance, we tried letting $\pi_v^0 = 0$ for all sources v, setting

$$\pi_w^0 = \max \{ c_{vw} \mid (v, w) \in \mathcal{L} \}$$

for all sinks w, and using a "greedy" heuristic to set $\mathbf{x}^0$ (or $\mathbf{y}^0$). Such procedures appeared to have no beneficial effects on the convergence of the method.

We did, however, identify an initialization procedure that produced a tangible and consistent, if unspectacular (about 25%) reduction in the number of iterations needed for convergence. This procedure involves noticing, as we have already mentioned, that large values of λ allow large dual steps, whereas small values of λ promote large primal steps. The idea is, at the outset of the algorithm, to use a small value of λ in the primal calculations, and a large value for the dual ones, thus inducing large primal steps *and* large dual steps.

More precisely, let $\{\lambda_x(k)\}_{k=0}^{\infty}$ be a positive, nondecreasing sequence and $\{\lambda_\pi(k)\}_{k=0}^{\infty}$ be a positive, nonincreasing sequence such that for some $K \geq 0$, $k \geq K$ implies that $\lambda_x(k) = \lambda_\pi(k) = \lambda$. We then execute the following iteration, starting with $\mathbf{y}^0 = \mathbf{0}$ and $\pi^0 = \mathbf{0}$:

$$x_j^{k+1} = P_{V(j)}\left[y_j^k + \frac{1}{\|\mathbf{a}_j\|^2}\left(\left[\sum_{i=1}^{m} \frac{a_{ij} r_i(\mathbf{y}^k)}{q_i}\right] - \frac{\bar{c}_j(\pi^k)}{\lambda_x(k)}\right)\right] \qquad j=1,\ldots,n$$

$$y_j^{k+1} = (1-\rho_k) y_j^k + \rho_k x_j^{k+1} \qquad j=1,\ldots,n \qquad \text{(TLH)}$$

$$\pi_i^{k+1} = \pi_i^k + \frac{\lambda_\pi(k)\rho_k}{q_i} r_i(\mathbf{x}^{k+1}) \qquad i=1,\ldots,m \ .$$

For all $k \geq K$, this iteration is identical to the generalized alternating step method. We may thus consider the first K iterations of (TLH) to be a heuristic for setting the initial values $\mathbf{y}^K$ and π^K to be used by the generalized alternating step method.

Now, there are infinitely many possible choices for the sequences $\{\lambda_x(k)\}_{k=0}^{\infty}$ and $\{\lambda_\pi(k)\}_{k=0}^{\infty}$. After extensive testing, we determined that the following strategy seemed to work well: one starts with $\lambda_x(0) = \lambda/10$, increasing $\lambda_x(k)$ by a factor of 1.05 once every 10 iterations until $\lambda_x(k)$ reaches λ. Similarly, $\lambda_\pi(0)$ is set to 10λ, and $\lambda_\pi(k)$ is reduced by a factor of 0.90 once every 5 iteration until it reaches λ. Formally,

$$\begin{aligned}
\lambda_x(0) &= \frac{\lambda}{10} \\
\lambda_x(k+1) &= \begin{cases} \min\{\lambda, (1.05)\lambda_x(k)\}, & k+1 \equiv 0 \pmod{10} \\ \lambda_x(k) & \text{otherwise} \end{cases} \\
\lambda_\pi(0) &= 10\lambda \\
\lambda_\pi(k+1) &= \begin{cases} \max\{\lambda, (0.9)\lambda_\pi(k)\}, & k+1 \equiv 0 \pmod{5} \\ \lambda_\pi(k) & \text{otherwise} \end{cases} .
\end{aligned}$$

We call this choice of $\{\lambda_x(k)\}_{k=0}^{\infty}$ and $\{\lambda_\pi(k)\}_{k=0}^{\infty}$, in conjunction with the iteration (TLH), the *twin lambda heuristic*. Note that there is no convergence theory for the iteration (TLH) in the event $\lambda_x(k)$ and $\lambda_\pi(k)$ do not eventually become equal. Empirical tests seem to confirm that the iteration will generally *not* converge if

$$\lim_{k \to \infty} \lambda_x(k) \; < \; \lim_{k \to \infty} \lambda_\pi(k) \;\; ,$$

although swift convergence may occur for individual problem instances.

Table 4 summarizes experience with the twin lambda heuristic on the same 22 test problems used in Table 3. Again, the "average number of iterations" column gives the mean number of iterations, over the 22 problems, to reduce $\bar{r}(k)$ and $\varepsilon(k)$ to 0.001. The parameter λ was set to θc_{max}, where θ is given by the table. We used over-relaxation by a constant factor ρ, also given in the table.

θ	ρ	**Average Number of Iterations**
0.15	1.50	539
0.15	1.00	416
0.10	1.00	350

Table 4. Experience with the twin lambda heuristic for the generalized alternating step method.

With $\theta = 0.15$ and $\rho = 1.5$, the twin lambda heuristic actually degrades the performance of the algorithm by a small amount, raising the average number of iterations from 469 (the best result Table 3) to 539. When the relaxation factor ρ is set back to 1, however, performance improves to 416 average iterations, better than any entry in Table 3. Decreasing θ to 0.10 gives a further reduction to 350 average iterations, or about 25% better than the best result obtained without the twin lambda heuristic. In fact, using

$\theta = 0.1$ and the twin lambda heuristic without over- or under-relaxation is the most efficient variant of the alternating step method so far identified for assignment problems.

Table 5 details the performance of this variant of the alternating step method on all 22 problems in the data set of Tables 3 and 4. All runs were terminated when both $\bar{r}(k)$ and $\varepsilon(k)$ were 0.001 or less.

The CPU times in the table are for an Alliant FX/8 computer, using four of the possible eight processors and pipelined vector arithmetic. The program, designed exclusively for network problems, was written in FX/FORTRAN under the CONCENTRIX operating system. All quantities in the "iterations" column are multiples of 10 because $\bar{r}(k)$ and $\varepsilon(k)$ were only computed every 10 iterations. The termination test was done in this periodic manner because the test is about as time-consuming as the iteration itself. The "exact" column contains a check mark whenever, on termination, $\bar{r}(k)$ and $\varepsilon(k)$ were both zero to within machine precision (using the standard single-precision IEEE floating point representation), as opposed to merely less than 0.001. Note that this exact convergence occurs in the majority of cases. Thus, despite the negative results of Section 6.2, the algorithm does have a distinct tendency to "jump" to an exact solution when it gets close enough to one. The "early" column is checked for all problems were convergence occurred before the end of the twin lambda heuristic, that is, before $\lambda_x(k) = \lambda_\pi(k) = \lambda$. All the test problems were created with NETGEN, and had integer arc costs uniformly distributed between 1 and c_{max}.

We also tried the alternating step method on some large, completely dense (1024 × 1024 node, 1,048,576 arc) assignment problems. Here, the results were much less

Problem	Sources/ Sinks	Arcs	c_{max}	Iterations	CPU Seconds	Exact	Early
1	4	16	982	60	0.02	✓	✓
2	32	100	100	60	0.03	✓	✓
3	32	400	100	110	0.13	✓	✓
4	32	600	100	1020	1.47		
5	32	800	100	770	1.40		
6	32	900	100	350	0.70		
7	32	1024	100	60	0.15	✓	✓
8	32	1024	100	50	0.12	✓	✓
9	32	1024	100	60	0.13	✓	✓
10	1024	5120	100	440	7.13	✓	✓
11	1024	5120	100	770	12.57	✓	
12	1024	5120	100	930	15.52	✓	
13	1024	5120	100	800	13.23	✓	
14	200	1000	100	180	0.58	✓	✓
15	400	2000	100	390	2.48	✓	✓
16	600	3000	100	430	4.07	✓	✓
17	100	500	100	490	0.83		
18	105	525	200	140	0.25	✓	✓
19	110	550	400	160	0.32	✓	✓
20	115	575	800	140	0.27	✓	✓
21	129	600	1600	140	0.28	✓	✓
22	125	625	3200	150	0.32	✓	✓

Table 5. Detailed results for the alternating step method without over- or under-relaxation, using $\theta = 0.1$ and the twin lambda heuristic.

satisfactory; in fact, the method converged to slowly to run any complete tests. The apparent reason for this poor performance is that the node degrees $d(v)$ and $d(w)$, hence

the parameters q_i, were so large that the method could make only miniscule steps. Thus, it appears that for problems with large numbers of nodes, the alternating step method requires a certain minimum level of sparsity to be effective.

Still, the content of Table 5 is moderately encouraging. For all but a few harder problems (4, 5, 6, and 17), the algorithm converges exactly in a number of iterations roughly equal to, or less than, the total number of nodes. The run times are several orders of magnitude slower than those that could be obtained by state-of-the-art assignment codes such as AUCTION (Bertsekas 1979b, 1985, 1988, Bertsekas and Eckstein 1987, 1988), because each iteration requires the processing of every arc in the problem network. However, the alternating step method is highly parallelizable, so the relatively low iteration counts shown hold some promise that the method might be competitive, given sufficiently massive parallelism.

7.2.5. Other Variations on the Method

We tried several other modifications of the alternating step method, none of which seemed consistently beneficial. One of these variations was to apply a Gram-Schmidt procedure to the rows of the constraint matrix **A**, producing an equivalent constraint matrix $\tilde{\mathbf{A}}$ with all rows orthogonal, and redundant constraints eliminated. The principal effect of this procedure was to destroy sparsity, and thus increase many of the constraint degrees $d(i)$. Larger $d(i)$ mean that the alternating step method must take smaller steps, resulting in slower convergence. The less drastic procedure of normalizing the rows of **A** so that $\| \mathbf{A}_i \|_2 = 1$ for all i had little effect on run times.

We also tried an algorithm that is a hybrid between the alternating direction method of multipliers and the conventional method of multipliers, namely

do L ***times***

begin

$$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}'} \{ f(\mathbf{x}') + \langle \mathbf{p}, \mathbf{M}\mathbf{x}' \rangle + \frac{\lambda}{2} \| \mathbf{M}\mathbf{x}' - \mathbf{z} \|^2 \} ;$$

$$\mathbf{z} \leftarrow \arg\min_{\mathbf{z}'} \{ g(\mathbf{z}') - \langle \mathbf{p}, \mathbf{z}' \rangle + \frac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z}' \|^2 \} ;$$

end;

$$\mathbf{p} \leftarrow \mathbf{p} + \lambda(\mathbf{M}\mathbf{x} - \mathbf{z}); \quad .$$

When $L = 1$, this method is equivalent to the alternating direction method of multipliers. When L is very large, the alternating $\mathbf{x}'$ and $\mathbf{z}'$ optimizations effectively minimize the augmented Lagrangian $f(\mathbf{x}) + g(\mathbf{z}) + \langle \mathbf{p}, \mathbf{M}\mathbf{x} - \mathbf{z} \rangle + \frac{\lambda}{2} \| \mathbf{M}\mathbf{x} - \mathbf{z} \|^2$ in both $\mathbf{x}$ and $\mathbf{z}$, so the algorithm is essentially the method of multipliers. For intermediate values of L, one obtains a new method, which we call the *truncated method of multipliers*, for which there is no established convergence theory.

Applying the truncated method of multipliers with the choices of f, g, and $\mathbf{M}$ that yield the alternating step method, we found no real computational advantage for assignment problems (or minimum cost flow problems in general, for that matter). For intermediate values of L, the number of primal (that is, $\mathbf{x}$ and $\mathbf{z}$) updates needed to obtain convergence seemed roughly the same as for the alternating step method. A more thorough investigation could certainly be made, but the approach did not seem very promising.

7.3. Comparison with the Alternating Direction Transportation Algorithm

We also compared the alternating step method for linear programming with a related method designed specifically for transportation problems, which we call the *alternating*

direction transportation algorithm. This method is derived in Exercise 5.3.10 of Bertsekas and Tsitsiklis (1989). We will now give a brief description of this algorithm, and sketch one possible derivation.

We first generalize the formulation of the assignment problem (AP) to a formulation for the transportation problem. We still let $\mathcal{V}$ and $\mathcal{W}$ be finite sets of sources and sinks, respectively, but no longer require that they have the same number of elements. We let $\{a_v\}_{v \in \mathcal{V}}$ and $\{b_w\}_{w \in \mathcal{W}}$ be sets of positive integer supplies and demands such that $\sum_{v \in \mathcal{V}} a_v = \sum_{w \in \mathcal{W}} b_w$. Again, $\mathcal{L} \subseteq \mathcal{V} \times \mathcal{W}$ is the set of arcs. The formulation of the transportation problem is then

$$\begin{aligned} \text{minimize} \quad & \sum_{(v,w) \in \mathcal{L}} c_{vw} x_{vw} \\ \text{subject to} \quad & \sum_{w:\,(v,w) \in \mathcal{L}} x_{vw} = a_v \qquad \forall\ v \in \mathcal{V} \\ & \sum_{v:\,(v,w) \in \mathcal{L}} x_{vw} = b_w \qquad \forall\ w \in \mathcal{W} \\ & x_{vw} \geq 0 \qquad\qquad\qquad \forall\ (v,w) \in \mathcal{L} \quad . \end{aligned} \tag{TP}$$

We first convert this problem to our standard form (P),

$$\text{minimize}_{\ \mathbf{x} \in \mathbb{R}^n}\ f(\mathbf{x}) + g(\mathbf{M}\mathbf{x}) \quad ,$$

by setting

$$f(\mathbf{x}) \ = \ \begin{cases} \displaystyle\sum_{(v,w) \in \mathcal{L}} c_{vw} x_{vw} \ , & \displaystyle\sum_{w:\,(v,w) \in \mathcal{L}} x_{vw} = a_v \ \forall\, v \in \mathcal{V},\ x_{vw} \geq 0 \ \forall\, (v,w) \in \mathcal{L} \\ +\infty \ , & \text{otherwise} \end{cases}$$

$$g(\mathbf{z}) \;=\; \begin{cases} 0\,, & \displaystyle\sum_{v:\,(v,w)\in\mathcal{L}} x_{vw} \;=\; b_w \quad \forall\; w \in \mathcal{W} \\ +\infty\,, & \text{otherwise} \end{cases}$$

$$\mathbf{M} \;=\; \mathbf{I} \qquad (\text{the } |\mathcal{L}| \times |\mathcal{L}| \text{ identity matrix}) \quad .$$

It is now possible to apply the alternating direction method of multipliers to this formulation. We omit the details of the derivation, and simply give the final algorithm that results. Let $l = |\mathcal{L}|$. The algorithm generates a sequence of flows $\{\mathbf{x}^k\}_{k=0}^{\infty} \in \mathbb{R}^l$ and a sequence of dual variables $\{\pi^k\}_{k=0}^{\infty} \in \mathbb{R}^{|\mathcal{W}|}$, one dual variable $\pi_w{}^k$ associated with each sink node w. For any vector $\mathbf{x} \in \mathbb{R}^l$, let

$$r_w(\mathbf{x}) \;=\; \left(\sum_{v:\,(v,w)\in\mathcal{L}} x_{vw} \right) - b_w$$

for all sinks w. Let $d(w)$, for all $w \in \mathcal{W}$, denote the degree of sink w in the problem network, and let $d(v)$ denote the degree of source node v for all sources $v \in \mathcal{V}$. Given any k, let $\mathbf{x}_v{}^k$ denote the subvector of $\mathbf{x}^k$ with components $\{x_{vw}{}^k\}$ for all w such that $(v, w) \in \mathcal{L}$. We also define an auxiliary sequence $\{\mathbf{u}^k\}_{k=0}^{\infty} \in \mathbb{R}^l$, with a similar indexing scheme to $\mathbf{x}^k$. In general, let "Project_Simplex(d, a, $\mathbf{u}$)" denote the projection of $\mathbf{u} \in \mathbb{R}^d$ onto the d-dimensional simplex

$$S(d, a) \;=\; \{\; \mathbf{q} \in \mathbb{R}^d \;|\; q_1 + \ldots + q_d \;=\; a \;\geq\; 0,\; \mathbf{q} \geq \mathbf{0} \;\} \quad .$$

Then, the alternating direction method of multipliers for the above choice of f, g, and $\mathbf{M}$ turns out to reduce to

$$
\begin{aligned}
u_{vw}^{k+1} &= x_{vw}^{k+1} - \frac{r_w(\mathbf{x}^k)}{d(w)} + \frac{1}{\lambda}(c_{vw} + \pi_w^k) && \forall\ (v, w) \in \mathcal{L} \\
\mathbf{x}_v^{k+1} &= \text{Project\_Simplex}(d(v), a_v, \mathbf{u}_v^{k+1}) && \forall\ v \in \mathcal{V} \\
\pi_w^{k+1} &= \pi_w^k + \frac{\lambda}{d(w)} r_w(\mathbf{x}^{k+1}) && \forall\ w \in \mathcal{W}\ ,
\end{aligned}
\qquad \text{(ADT)}
$$

where $\mathbf{x}^0 \in \mathbb{R}^l$ and $\pi^0 \in \mathbb{R}^{|\mathcal{W}|}$ are arbitrary. This method is the alternating direction transportation algorithm. It can also be derived by specializing the method (BSLCM) mentioned at the end of Chapter 5.

We coded a version of the alternating direction transportation algorithm in FX/FORTRAN on the Alliant FX/8 computer. Table 6 presents a comparison of the alternating direction transportation algorithm and the alternating step method for linear programming for the first 16 assignment problems of Table 5. The methods were run with identical parameters, using the twin lambda heuristic. To use the twin lambda heuristic, the alternating direction transportation algorithm was modified to

$$
\begin{aligned}
u_{vw}^{k+1} &= x_{vw}^{k+1} - \frac{r_w(\mathbf{x}^k)}{d(w)} + \frac{1}{\lambda_x(k)}(c_{vw} + \pi_w^k) && \forall\ (v, w) \in \mathcal{L} \\
\mathbf{x}_v^{k+1} &= \text{Project\_Simplex}(d(v), a_v, \mathbf{u}_v^{k+1}) && \forall\ v \in \mathcal{V} \\
\pi_w^{k+1} &= \pi_w^k + \frac{\lambda_\pi(k)}{d(w)} r_w(\mathbf{x}^{k+1}) && \forall\ w \in \mathcal{W}\ ,
\end{aligned}
$$

where $\{\lambda_x(k)\}_{k=0}^{\infty}$ and $\{\lambda_\pi(k)\}_{k=0}^{\infty}$ are as described in Section 7.2.4. Check marks indicate indicate cases in which the transportation algorithm converged to machine precision; the remaining problems were terminated when the total flow into all sinks w was within 0.001 of b_w, and 0.001-complementary slackness was met.

Problem	Sources/ Sinks	Arcs	*Alternating Step Method* Iterations	CPU Seconds	*Alternating Direction Transportation Algorithm* Iterations		CPU Seconds
1	4	16	60	0.02	10	✓	0.06
2	32	100	60	0.03	20	✓	0.25
3	32	400	110	0.13	50	✓	0.63
4	32	600	1020	1.47	280		1.27
5	32	800	770	1.40	420		1.90
6	32	900	350	0.70	220		1.62
7	32	1024	60	0.15	40	✓	1.40
8	32	1024	50	0.12	30	✓	1.38
9	32	1024	60	0.13	40	✓	1.41
10	1024	5120	440	7.13	230	✓	12.15
11	1024	5120	770	12.57	280	✓	12.88
12	1024	5120	930	15.52	400	✓	14.67
13	1024	5120	800	13.23	260	✓	12.57
14	200	1000	180	0.58	120	✓	2.07
15	400	2000	390	2.48	250	✓	4.87
16	600	3000	430	4.07	220	✓	7.02

Table 6. Comparison of the alternating step method and the alternating direction transportation algorithm on a set of 16 assignment problems. Check marks indicate problems for which the alternating direction transportation algorithm converged to machine precision.

The reader will notice a striking correlation between the number of iterations needed by the two methods. Indeed, the alternating transportation algorithm seems to take roughly half as many iterations as the alternating step method in nearly all cases, as illustrated in Figure 26. A comparison of actual run times, however, reveals that the alternating step

method is generally somewhat faster, because the calculations needed for each iteration are less complicated. The run-time advantage of the alternating step method is particularly large for the easier problems, and is less pronounced (or even absent) for problems requiring a large number of iterations. This effect is due to a trick used in the implementation of the "Project_Simplex" subroutine of the transportation algorithm. Essentially, the code was designed to run most quickly when $\mathbf{u}_V^{k+1}$ and $\mathbf{u}_V^{k}$ are almost equal.

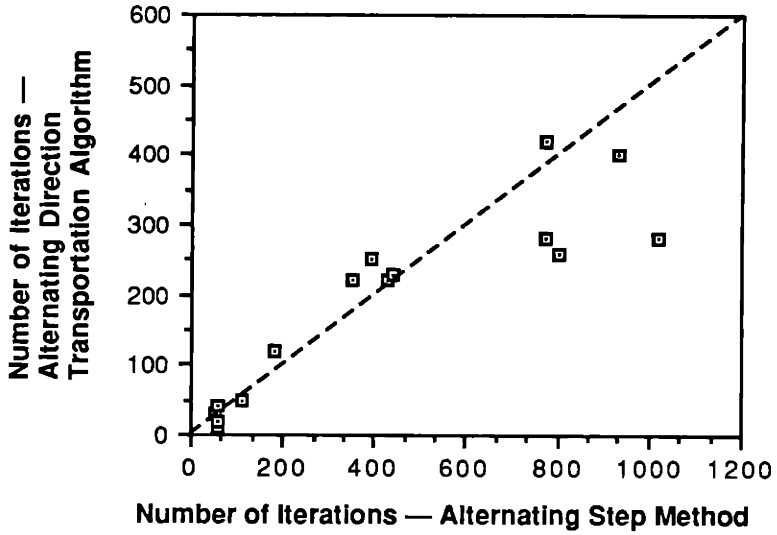


Figure 26. Comparison of the number of iterations required for assignment problems by the alternating step method and the alternating direction transportation algorithm.

Also, note that the cases in which the transportation method gives exact convergence are precisely the same as those for which the alternating step method converges exactly (compare Table 6 with Table 5). Apart from the two methods' having very similar derivations, it is presently unknown why their performance should correlate quite so strongly. It does not appear at the moment that the alternating direction transportation

algorithm offers any substantial advantages over the alternating step method, and it is significantly more complex to code.

7.4. Experience with Transportation Problems

While the alternating step method converges in a fairly promising manner for assignment problems, it does not appear to work well on more general linear programs. We will now illustrate this phenomenon in the case of transportation problems.

We ran the alternating step method for linear programming on a sequence of 8 NETGEN-generated transportation problems of the form (TP) given in Section 7.3,

$$\begin{aligned} \text{minimize} \quad & \sum_{(v,w) \in \mathcal{L}} c_{vw} x_{vw} \\ \text{subject to} \quad & \sum_{w:\,(v,w) \in \mathcal{L}} x_{vw} = a_v \qquad \forall\ v \in \mathcal{V} \\ & \sum_{v:\,(v,w) \in \mathcal{L}} x_{vw} = b_w \qquad \forall\ w \in \mathcal{W} \\ & x_{vw} \geq 0 \qquad\qquad\qquad \forall\ (v,w) \in \mathcal{L} \quad . \end{aligned} \tag{TP}$$

In all cases, there were 100 sources, 100 sinks, and 750 arcs. The supplies $\{a_v\}_{v=1}^{100}$ and demands $\{b_w\}_{v=1}^{100}$ were all positive integers; over the sequence of problems, the average value of the a_v (which was necessarily also the average value of the b_w, since the number of sources and sinks were equal) was gradually increased from 1 to 20. Note that an average supply of 1 yields an assignment problem. Table 7 shows the number of iterations needed to reduce $\bar{r}(k)$ and $\varepsilon(k)$ to 0.001, along with similar results for the alternating direction transportation algorithm. For the alternating step method, we introduced the redundant upper bound constraints

$$x_{vw} \leq \min\{a_v, b_w\} \quad \forall\ (v, w) \in \mathcal{L}\ ,$$

which greatly improved performance. The standard choice of $\lambda = (0.1)c_{\max}$ and the twin lambda heuristic were used for both algorithms.

Average Supply	Alternating Step Method Iterations	Alternating Direction Transportation Algorithm Iterations
1	120	100
1.5	180	200
2	1140	770
3	1930	1140
5	4710	2690
8	More than 5000	3420
10	More than 5000	2590
20	More than 5000	4270

Table 7. Performance of the alternating step method and alternating direction transportation algorithm on a sequence of transportation problems of gradually increasing average supply and demand.

Figure 27 displays the data from Table 7. As is evident from the figure, the number of iterations grows very rapidly indeed with the average supply. The alternating direction transportation algorithm results are similar, if less dramatic. The reasons for this extreme performance degradation in response to increasing average supply are presently unknown. The results *are* in accordance with the those of Section 6.3.4, which gives that the convergence rate bound $\tau = \alpha/\sqrt{\alpha^2 + 1}$, where

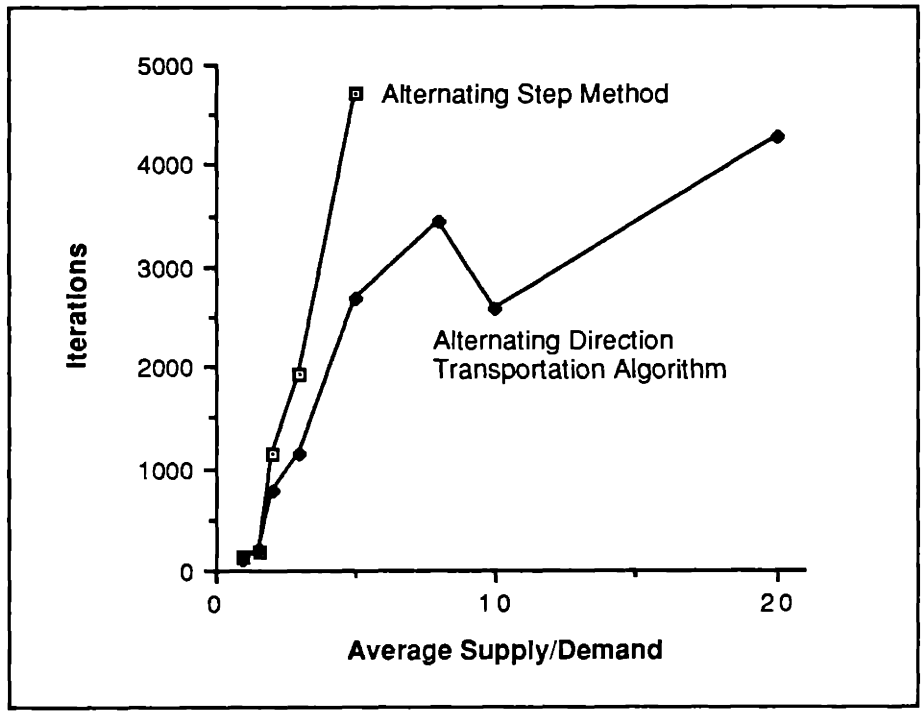


Figure 27. Growth in the number of iterations in response to average node supply/demand in transportations problems.

$$\alpha = \left(\sqrt{mn} + \lambda \left[\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 \right]^{1/2} \right) (m+n)\Delta(\mathbf{Z}) \left[a_{\max} R_x + \frac{d_{\max}}{\lambda} R_\pi \right] \quad .$$

Recalling that

$$\mathbf{Z} = \begin{bmatrix} -\mathbf{I} & 0 \\ \mathbf{A} & 0 \\ -\mathbf{A} & 0 \\ 0 & \mathbf{A}^T \\ \mathbf{c}^T & -\mathbf{b}^T \end{bmatrix} ,$$

we see that $\Delta(\mathbf{Z})$ can increase explosively as the magnitude of the right-hand side vector $\mathbf{b}$ grows. This growth in $\Delta(\mathbf{Z})$ would cause τ to rapidly approach 1. However, the same

kind of reasoning would suggest a similar slowing of the convergence rate as the magnitude of $\mathbf{c}$ grows, which is *not* observed in practice for the proper choice of λ. Altering the choice of λ from $(0.1)c_{\max}$ did not improve convergence for transportation problems.

In an attempt to improve transportation problem convergence, we tried a number of modifications to the alternating step method, but none had a consistent beneficial effect. The simplest strategy was to scale each constraint so that all the right-hand side entries were 1. This technique appeared only to worsen the convergence of the algorithm. Normalizing and/or orthogonalizing the rows of the constraint matrix $\mathbf{A}$ was also fruitless. We also tried a number of more obscure schemes involving dynamically changing λ or the relaxation parameter ρ_k of the generalized algorithm. These ploys met with success in isolated problems, but in general were not helpful.

The basic difficulty appears to be a persistent "spiralling" of the type mentioned in Section 7.1. Accordingly, we attempted a modification of the algorithm whereby $\mathbf{x}^k$ and π^k were periodically replaced by averages taken over a large number of prior iterations. This approach was extremely successful for very small problems, but, curiously, was not effective for larger ones.

In conclusion, we do not know how to effectively apply the alternating step method to linear programs more complicated than assignment problems. It is possible that some kind of scaling or constraint matrix preconditioning procedure might be effective, but this topic must be left for future research.

7.5. Experiences on Particular Parallel Computers

In this final section, we will discuss experience with the alternating step method on commercially available parallel processor systems. We concentrate exclusively on assignment problems. The alternating step method has potential to be a competitive method for assignment problems, but only if parallel speedups on the order of hundreds or thousands, which are conceivable for large problems, are attained. We have not yet obtained speedups of more than about 16, as we shall presently explain.

We will discuss three computer systems: the Alliant FX/8, the DAP-510, and the Connection Machine 2. Time on all these computers was generously donated by the Advanced Computing Research Facility (ACRF) of the Argonne National Laboratory. Professor Sandy Pentland of the MIT Media Lab also made Connection Machine 2 time available.

7.5.1. The Alliant FX/8

The Alliant FX/8 is a general purpose, shared memory, multiple instruction, multiple data (MIMD) multiprocessor containing, when fully configured, 8 numerical processing units. Each of these processors has a "pipelined" arithmetic unit capable of 32-operand vector operations. The vector mode of the each arithmetic unit, while not in itself truly parallel, can run almost 4 times as fast as the scalar, one-operand mode. Thus, using all 8 processors in vector mode, the system can theoretically run about 30 times faster than when a single processor using only the scalar mode. We found that for the alternating step method for assignment problems, we could achieve speedups on the order of 12 to 16.

After some initial development on Apple Macintosh personal computers and Digital Equipment Corporation VAXstation 2000 workstations, we performed the bulk of our computational testing on the Alliant. The Alliant's FX/FORTRAN compiler required fairly minimal program modifications to create a version of the alternating step method that took advantage of both multiple processors and vector arithmetic.

We coded two principal version of the alternating step method on the Alliant FX/8: a general linear programming implementation using a dense matrix representation, and a specialized version for network problems, using a sparse representation. The Alliant CONCENTRIX operating system has a feature that allows one to specify the number of processors, from 1 to 8, to be used to execute a program. Table 8 gives the results of running our dense implementation on a 32×32 node, 400 arc assignment problem with between 1 and 8 processors, using vector arithmetic.

Number of Processors	Run Time (Seconds)	Speedup	Efficiency
1	49.47	1.00	1.00
2	24.95	1.98	0.99
3	17.15	2.83	0.94
4	12.57	3.94	0.98
5	10.05	4.92	0.98
6	8.52	5.81	0.97
7	7.53	6.57	0.94
8	6.33	7.82	0.98

Table 8. Performance of a dense Alliant FX/8 implementation of the alternating step method on a 32×32 node, 400 arc assignment problem, with $\lambda = (0.1)c_{\max}$ and without the twin lambda heuristic. These runs used vector arithmetic; run time for one processor without vector arithmetic was 102.90 seconds.

Notice that all the efficiencies in Table 8 are at least 94%. The maximum speedup attained by using all 8 processor was approximately 7.8. However, a one-processor run without vector arithmetic required 102.90 seconds, or about 2.1 times as long as a single-processor run *with* vector arithmetic. Thus the total speedup attributable to both vector arithmetic and multiprocessing was approximately $2.1 \times 7.8 \approx 16.4$.

Still, the minimum time of 6.33 seconds is hardly impressive for an assignment problem of such small size. Using our specialized network version of the alternating step method in conjunction with the twin lambda heuristic, the time for the problem can be reduced to 0.13 seconds, using only 4 processors (see Table 5). To accurately measure the speedup attributable to parallelism for our specialized network code, we decided to use a larger assignment problem, with 1024 sources, 1024 sinks, and 5120 arcs. The results are given in Table 9.

Number of Processors	Run Time (Seconds)	Speedup	Efficiency
1	52.82	1.00	1.00
2	26.48	1.99	0.99
3	18.48	2.86	0.95
4	13.80	3.83	0.96
5	11.53	4.58	0.92
6	9.37	5.64	0.94
7	8.23	6.42	0.92
8	7.52	7.02	0.88

Table 9. Performance of the Alliant FX/8 network implementation on a 1024 × 1024 node, 5120 arc assignment problem, with $\lambda = (0.1)c_{\max}$ and the twin lambda heuristic. These runs used vector arithmetic; run time for one processor without vector arithmetic was 88.60 seconds.

All efficiencies in Table 9 are at least 88%, and the maximum speedup attributable to parallelism was approximately 7.0. A one-processor run without vector arithmetic took 88.60 seconds, or roughly 1.7 times longer than a single-processor vector run; thus the total speedup attributable to parallelism and vectorization is about $7.0 \times 1.7 \approx 11.9$.

In conclusion, the alternating step method appears to be able to make very efficient use of the small number of processors available on the Alliant; using 8 processors yields a speedup of at least a factor of 7. The potential of vector arithmetic is not as fully exploited, but it still speeds up the algorithm by roughly a factor of 2. Unfortunately, as can be seen from the run times in Table 9, the alternating step method is still far from being competitive with state-of-the-art assignment codes such as AUCTION. To achieve such competitiveness, one must employ enough processors to yield speedups on the order of hundreds or thousands. Clearly, such speedups require a more massively parallel system than the Alliant FX/8.

7.5.2. The Active Memory Technologies DAP-510

The Active Memory Technologies DAP-510 is a specialized, British-designed array processor with an unusual memory organization. It consists of of 1024 one-bit processors, arranged in a 32×32 grid, each with 8K bytes of local memory. The architecture is SIMD (single instruction, multiple data), with a provision to "mask off" subsets of processors. The DAP-510's main advantage is that some its microcoded vector and matrix operations are extremely fast. For example, it can find the maximum element from a list of 1024 integers in about the same time that it takes to add two numbers. The DAP architecture also permits rapid broadcast of information along rows or columns of the processor grid. The main deficiency of the architecture is that communi-

cation other than row and column broadcasting can be very slow, and must be done in an essentially serial manner by the Motorola 68020 microprocessor that controls the grid. The DAP is also somewhat slow at floating-point arithmetic.

Most instructions on the DAP have a scalar (single-operand), vector (32 parallel operand) and matrix (1024 parallel operand) modes. We experimentally determined that for floating-point data, matrix instructions run an average of 280 times faster than 1024 equivalent scalar instructions. Thus, the DAP-510 is theoretically capable of speedups on the order of hundreds.

We programmed the DAP in FORTRAN PLUS, an extension of FORTRAN 77 containing special functions and datatypes for 32-element vectors and 32×32-element arrays.

Originally, we had thought that the DAP would be well suited to the alternating step method, but this turned out not to be the case. The reason is that, because of its rigid communications architecture, the DAP is suitable only for dense problems. As a first step, we implemented a simple FORTRAN PLUS code for 32×32 node, fully dense assignment problems. This implementation is described in Figure 28. One processor was assigned to each arc, the processor at grid position (v, w) handling the arc between source v and sink w. The main difficulty with this implementation is that the dual variable update steps are performed using the vector, 32-operand computation mode, hence the available speedup for these calculations was no more than 32. The price update steps thus become a bottleneck, limiting the speedup attainable for the algorithm as a whole.

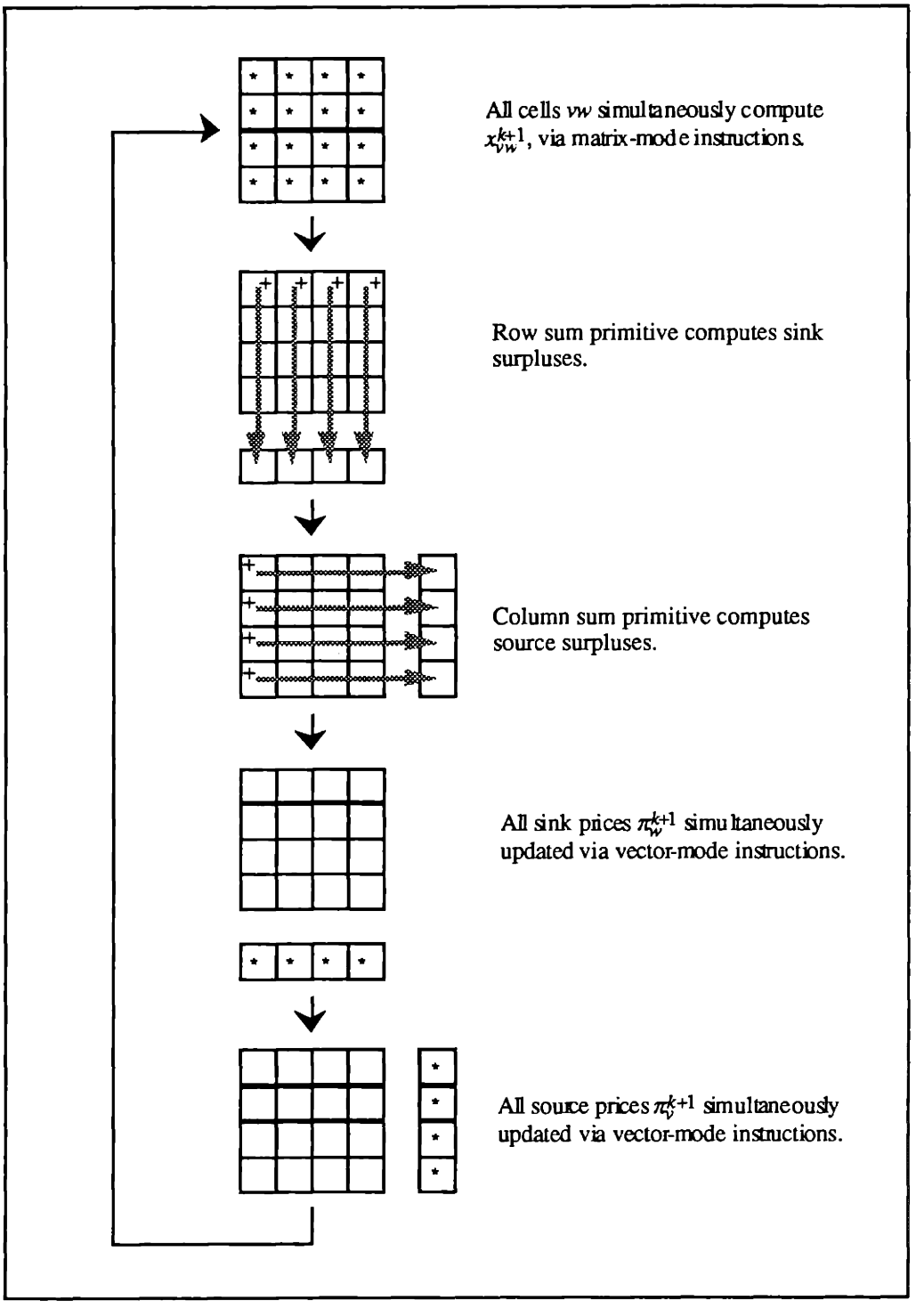


Figure 28. Schematic representation of a simple DAP implementation of the alternating step method for dense assignment problems. A 4×4 grid is shown; the actual dimension of the DAP processor grid is 32×32.

David Castañon then suggested an alternative implementation where, for dense 1024 × 1024 node problems, all the operations in the alternating step method iteration could be encoded using the DAP's matrix computation mode. We also implemented this approach, but found that the alternating step method, as mentioned earlier, does not appear to be suitable for dense problems of this size. For 1024 × 1024 node assignment problems with any substantial sparsity, this alternative DAP implementation was outperformed by many orders of magnitude by the Alliant network implementation.

In conclusion, the DAP does not appear to have a suitable architecture for the alternating step method for assignment problems. To obtain maximum efficiency, the DAP requires large, highly dense problems; it is precisely for these problems that the alternating step method fails to converge acceptably.

7.5.3. The Thinking Machines Connection Machine 2

The Thinking Machines *Connection Machine 2* (CM-2) computer is an specialized multiprocessor whose minimum configuration consists of 16,384 one-bit processors, arranged in a 14-dimensional hypercube. Each processor has 64K bits of local memory. Each group of 32 consecutively-numbered processors shares a pipelined floating-point arithmetic chip, while each set of 16 consecutively-numbered processors is connected to a specialized "router" chip. The router chips, 1024 in all, form a 10-dimensional hypercube communication network. The hardware and firmware of the CM-2 provide a set of extremely powerful (at least in appearance) communications primitives that permit concurrent interprocessor communication in arbitrary patterns. The CM-2's architecture, like the DAP's, is SIMD (single instruction, multiple data), with the option of "turning off" subsets of the processors for some instructions.

The CM-2 is controlled by a Symbolics, Sun, or Digital Equipment VAX "front end" computer, which must perform surprisingly low-level tasks, such as fetching individual CM-2 macroinstructions. The runs described in this section were done on the Argonne National Laboratory CM-2, which has a Sun-4 front end processor. The Sun-4 appears to be the most efficient front end available.

At the time of writing, there were four principal programming environments for the CM-2: C/PARIS, LISP/PARIS, C*, and *LISP. C/PARIS and LISP/PARIS are essentially CM-2 machine language instruction sets grafted onto the C and LISP languages of the front end processor, respectively. C* is a *bona fide* high level language, which extends C by providing data structures called *domains*, each instance of which resides on a different CM-2 processor. *LISP is a Common LISP macro package that gives LISP programmers low-level access to the CM-2 hardware more conveniently than the PARIS environments. Only pre-release versions of CM-2 FORTRAN were available at the time that this research was performed; we did not attempt to use them.

The general architecture of the CM-2 seems better suited to the alternating step method than that of the DAP, especially for sparse network problems. The ability of the CM-2 to send messages in arbitrary patterns means that processors can theoretically be assigned to individual nodes and arcs of a sparse problem network without obvious waste or communications blockages.

We implemented two versions of the alternating step method on the CM-2, one in C* and one in *LISP. Both versions were intended only for network problems. The first implementation, in C*, was based closely on the bipartite implementation of Section 6.4.1. It

allocated one processor for each node, and one for each arc. We now recall the alternating step method for network problems:

$$x_{vw}^{k+1} = P_{V(vw)}\left[x_{vw}^{k} + \frac{1}{2}\left(\frac{r_v(\mathbf{x}^k)}{d(v)} - \frac{r_w(\mathbf{x}^k)}{d(w)} - \frac{\bar{c}_{vw}(\pi^k)}{\lambda}\right)\right] \qquad \forall \text{ arcs } (v,w)$$

$$\pi_v^{k+1} = \pi_v^k + \frac{\lambda}{d(v)} r_v(\mathbf{x}^{k+1}) \qquad \forall \text{ nodes } v \quad .$$

Following Section 6.4.1, we introduce auxiliary variables $\{\alpha_v{}^k\}_{k=0}^{\infty}$ for each node v, and rewrite the method as

$$x_{vw}^{k+1} = P_{V(vw)}\left[x_{vw}^{k} + \frac{1}{2}\left(\alpha_v^{k+1} - \alpha_w^{k+1} - \frac{c_{vw}}{\lambda}\right)\right] \qquad \forall \text{ arcs } (v,w)$$

$$\pi_v^{k+1} = \pi_v^k + \frac{\lambda}{d(v)} r_v(\mathbf{x}^{k+1}) \qquad \forall \text{ nodes } v$$

$$\alpha_v^{k+1} = \frac{r_v(\mathbf{x}^k)}{d(v)} - \frac{\pi_v^{k+1}}{\lambda} \qquad \forall \text{ nodes } v \quad .$$

Naturally, one can also introduce the twin lambda heuristic, obtaining

$$x_{vw}^{k+1} = P_{V(vw)}\left[x_{vw}^{k} + \frac{1}{2}\left(\alpha_v^{k+1} - \alpha_w^{k+1} - \frac{c_{vw}}{\lambda_x(k)}\right)\right] \qquad \forall \text{ arcs } (v,w)$$

$$\pi_v^{k+1} = \pi_v^k + \frac{\lambda_\pi(k)}{d(v)} r_v(\mathbf{x}^{k+1}) \qquad \forall \text{ nodes } v$$

$$\alpha_v^{k+1} = \frac{r_v(\mathbf{x}^k)}{d(v)} - \frac{\pi_v^{k+1}}{\lambda_x(k+1)} \qquad \forall \text{ nodes } v \quad .$$

The C* implementation proceeds as follows: The processor for each arc (v, w) first acquires the values of $\alpha_v{}^k$ and $\alpha_w{}^k$ from the processors assigned to its endpoint nodes v and w. Then, it computes $x_{vw}{}^{k+1}$ and sends messages to both v and w containing the result. All the processors that are assigned to arcs perform these operations simultaneously. The messages containing the new flow values accumulate at the node processors, and are added together upon arrival, yielding (with proper attention to signs)

the surpluses $r_v(\mathbf{x}^{k+1})$. The node processors simultaneously compute the new prices π_v^{k+1} and the auxiliary variables α_v^{k+1}, and the iteration repeats.

Unfortunately, the performance of the C* implementation was disappointing. For a 1024 × 1024 node, 5120 arc NETGEN assignment problem (problem 10 of Table 5), it required approximately 36 milliseconds per iteration. This timing compares unfavorably with the implementation on the Alliant, which needed only 16 milliseconds per iteration for the same problem. A close tracing of the C* code revealed that about 95% of the total run time was being consumed by communications functions. In the flow computation, all the processors for arcs incident to a given node v must retrieve the value of α_v^k from a single processor; similarly, in the recomputations of the surpluses, messages from all arcs incident to v must accumulate at one processor. Furthermore, C* makes it difficult to control exactly which processor will be assigned to a given task, so it is unlikely that the processor for node v and the processors for arcs incident to v will be close to one another in the CM-2's hypercube communication network. Thus, each iteration of the C* implementation creates a chaotic pattern of messages in which a large number of message collisions (that is, attempts by multiple messages to use the same communications link) are virtually assured. Such collisions have an adverse effect on CM-2 performance.

We therefore undertook to map the problem network onto the CM-2 processors in a more clever manner, using a technique developed by Zenios and Lasken (1988). In this approach, one allocates *two* processors to each arc (v, w), one associated with the "start" node v, and one associated with the "end" node w. Furthermore, one arranges the allocation so that all the processors associated with a given *node* have contiguous hypercube addresses (see Bertsekas and Tsitsiklis 1989, Chapter 1, for a description of hypercube addressing schemes). Each processor stores the flow of its associated arc and

the dual variable and surplus of its associated node. The alternating step method can then be implemented as follows, where the processor associated with arc (v, w) and node v is denoted (vw, v), and the processor associated with arc (v, w) and node w is denoted (vw, w):

(i) All processors (vw, v) send the value of α_v^k to (vw, w); simultaneously, all processors (vw, w) send the value of α_w^k to (vw, v).

(ii) All processors concurrently compute x_{vw}^{k+1} for their associated arc vw. Note that the computation is redundant, in that each arc flow is computed by two different processors.

(iii) Special CM-2 primitives called *segmented scans* are used to recompute the surpluses at all processors. It is in order to use these primitives that all processors associated with a given node must have consecutive hypercube addresses.

(iv) All processors update the dual variable π_v^{k+1} or π_w^{k+1} for their associated node. This computation is highly redundant, as the new dual variable for a each node v is computed separately at each processor associated with v. The number of such processors is equal to the degree of v.

The CM-2 communication pattern generated by this approach is much more desirable than that created by the C* implementation. There is one chaotic burst of communication in step (i), in which each processor sends and receives exactly one message. The only other communication activity is in the segmented scan operations of step (iii).

We implemented such an approach to the alternating step method in the most advanced CM-2 programming language that supports segmented scans, *LISP. Despite the front-end overhead incurred by LISP, this implementation was significantly faster than the C* version. The same iteration that required 36 milliseconds in C* took only 15 milliseconds, or slightly less time than on the Alliant. Of these 15 milliseconds, only about 6 milliseconds, or roughly 40%, appear to be absorbed by communication functions. A similar C/PARIS implementation would, in all likelihood, dramatically reduce the remaining 9 milliseconds of execution time by reducing overhead on the front end. We intend to write several C/PARIS versions of the alternating step method in the near future. As the *LISP implementation slightly outperforms the Alliant code, a C/PARIS versions should prove significantly faster than the Alliant.

Nevertheless, it is unlikely that such an implementation would prove that the alternating step method is a competitive method. The reason is as follows: for a sparse network, an iteration of the alternating step method requires only a handful of arithmetic operations for each arc or node. On modern computers, the time for such operations is measured in nanoseconds or microseconds, not milliseconds. Thus, even a few milliseconds of communication delay will slow the iteration by at least two orders of magnitude, largely offsetting any speedup due to parallelism. Thus, even though the results of Chapter 6 theoretically suggest that communication need not be a bottleneck in the operation of the alternating step method, it appears that communication overhead is in fact the overriding concern on current-generation multiprocessors such as the CM-2. To obtain a truly efficient, massively parallel implementation of the alternating step method, more attention will need to be paid to the details of communication. It is possible that the alternating step method might require a system with faster physical communication links and/or different low-level routing algorithms that the CM-2 provides.

Chapter 8

Conclusion

This thesis has shown that monotone operator splitting is a powerful, pervasive decomposition principle in mathematical programming. Splitting forms a common thread running through the gradient projection method and related algorithms for variational inequalities, the theory of the alternating direction method of multipliers (due to Glowinski, Marroco, Gabay, Fortin, and Le Tallec), the work of Spingarn, and the work of Gol'shtein. The Douglas-Rachford splitting scheme, as adapted by Lions and Mercier, is a particularly robust type of splitting that unifies the partial inverse work of Spingarn, the "general decompositional method" of Gol'shtein, and the alternating direction method of multipliers. Through the device of the splitting operator $S_{\lambda,A,B}$, we have tied the theory of Douglas-Rachford splitting to that of the proximal point algorithm. The insight gained in this manner has allowed a number of advances, including the *generalized* alternating direction method of multipliers, an improved understanding of the method of partial inverses, and the opportunity to apply theoretical results for the proximal point algorithm to new areas.

We have seen that the Douglas-Rachford type of splitting, by way of the (generalized) alternating direction method of multipliers, gives rise to new and intriguing convex programming decomposition algorithms, apparently suited to parallel implementation. These algorithms include the epigraphic projection method and the alternating step method for monotropic programming. We have paid particular attention to the alternating step method as applied to *linear* programming. In our study of the convergence rate of

that algorithm, the link between the theory of Douglas-Rachford splitting and that of the proximal point algorithm proved extremely useful.

Our computational tests of the alternating step method for linear programming have yielded mixed results: for sparse assignment problems, the method has shown some promise, but for more general problems, it seems to converge unacceptably slowly. There are already many efficient, highly specialized algorithms for the assignment problem; against such stiff competition, the alternating step method can only be successful if its potential for parallelism is fully realized. Because the method carries a relatively high communications penalty, finding a computer system and implementation approach that truly exploit this potential is not an easy task. It appears that the algorithm may require more powerful communication abilities than the current generation of commercial multiprocessors can provide. One topic for future research is the possibility of implementing an algorithm similar to the alternating step method on a "neural network" computing system. Tank and Hopfield (1986) have proposed neural networks to solve linear programs, but have given no proof of their validity.

Further research is needed to understand the convergence mechanism of the alternating step method. In particular, it would be helpful to understand why the performance of the algorithm degrades so rapidly when one moves from assignment problems to transportation problems. Barring progress on this front, the search for a truly efficient, massively parallel assignment implementation on a real multiprocessor should go forward.

It is interesting to note that no researcher working on Douglas-Rachford-related decomposition methods for classical optimization problems has ever documented truly encouraging computational results. The mildly favorable results given here, although

restricted to assignment problems, are in this sense something of a first. Perhaps an understanding of why the alternating step method works best on assignment problems will unlock the secret to improving the convergence of the Douglas-Rachford class of methods.

References

AGMON, S. (1954), "The Relaxation Method for Linear Inequalities". *Canadian Journal of Mathematics* **6**:382-392.

ARROW, K. J., HURWICZ, L., UZAWA, H., *ET. AL.* (1958), *Studies in Linear and Nonlinear Programming* (Stanford: Stanford University Press).

AUSLENDER, A. (1976), *Optimisation: Méthodes Numeriques* (Paris: Mason).

AUSLENDER, A. (1985), "Two General Methods for Computing Saddle Points with Applications for Decomposing Convex Programming Problems". *Applied Mathematics and Optimization* **13**(1):79-95.

BAKUSHINSKI, A. B., POLYAK, B. T. (1974), "On the Solution of Variational Inequalities". *Soviet Mathematics Doklady* **219**:1703-1710.

BENDERS, J. F. (1962), "Partitioning Procedures for Solving Mixed-Variables Programming Problems". *Numerische Mathematik* **4**:238-252.

BERTSEKAS, D. (1975), "Necessary and Sufficient Conditions for a Penalty Method to be Exact". *Mathematical Programming* **9**:87-99.

BERTSEKAS, D. (1976), "On the Goldstein-Levitin-Polyak Gradient Projection Method". *IEEE Transactions on Automatic Control* **21**(2):174-184.

BERTSEKAS, D. (1979A), "Convexification Procedures and Decomposition Methods for Nonconvex Optimization Problems". *Journal of Optimization Theory and Applications* **29**(2):169-197.

BERTSEKAS, D. (1979B), *A Distributed Algorithm for the Assignment Problem.* Unpublished report of the Laboratory for Information and Decision Sciences, MIT (available from the author).

BERTSEKAS, D. (1982), *Constrained Optimization and Lagrange Multiplier Methods* (New York: Academic Press).

BERTSEKAS, D. (1985), *A Distributed Asynchronous Relaxation Algorithm for the Assignment Problem*. Proceedings of the 24th IEEE Conference on Decision and Control, Fort Lauderdale, Florida, 1703-1704.

BERTSEKAS, D. (1986), *Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems*. Report LIDS-P-1606, Laboratory for Information and Decision Sciences, MIT. (Original Report, September 1986, revised version, November 1986.)

BERTSEKAS, D. (1988), "The Auction Algorithm: A Distributed Relaxation Method for the Assignment Problem". *Annals of Operations Research* **14**:105-123.

BERTSEKAS, D., ECKSTEIN, J. (1987), *Distributed Asynchronous Relaxation Methods for Linear Network Flow Problems*. Proceedings of International Federation of Automatic Control (IFAC), Munich.

BERTSEKAS, D., ECKSTEIN, J. (1988), "Dual Coordinate Step Methods for Linear Network Flow Problems". *Mathematical Programming* **42**(2):203-243.

BERTSEKAS, D., EL BAZ, D. (1987), "Distributed Asynchronous Methods for Convex Network Flow Problems". *SIAM Journal on Control and Optimization* **25**(1):74-85.

BERTSEKAS, D., GAFNI, E. M. (1982), "Projection Methods for Variational Inequalities with Application to the the Traffic Assignment Problem". *Mathematical Programming Study* **17**:139-159.

BERTSEKAS, D., TSITSIKLIS, J. (1989), *Parallel and Distributed Computation: Numerical Methods* (Englewood Cliffs: Prentice-Hall).

BOTSARIS, C. (1978A), "Differential Gradient Methods". *Journal of Mathematical Analysis and Applications* **63**(1):177-198.

BOTSARIS, C. (1978B), "A Class of Methods for Unconstrained Minimization Based on Stable Numerical Integration Techniques". *Journal of Mathematical Analysis and Applications* **63**(3):729-749.

BOTSARIS, C., JACOBSON, D. (1976), "A Newton-Type Curvilinear Search Method for Optimization". *Journal of Mathematical Analysis and Applications* **54**(1):217-229.

BREGMAN, L. M. (1965), "The Method of Successive Projection for Finding a Common Point of Convex Sets". *Soviet Mathematics Doklady* **6**(3):688-692.

BRÉZIS, H. (1971), "Monotonicity Methods in Hilbert Spaces and Some Applications to Nonlinear Partial Differential Equations", in *Contributions to Nonlinear Analysis*, E. H. ZARANTONELLO, editor (New York: Academic Press).

BRÉZIS, H., (1973), *Opérateurs Maximaux Monotones et Semi-Groupes de Contractions dans les Espaces de Hilbert* (Amsterdam: North-Holland, in French).

BRÉZIS, H., LIONS, P.-L. (1978), "Produits Infinis de Resolvantes". *Israel Journal of Mathematics* **29**(4):329-345 (in French).

BROWDER, F. E. (1964), "Continuity Properties of Monotone Nonlinear Operators in Banach Spaces". *Bulletin of the American Mathematical Society* **70**:551-553.

BROWDER, F. E. (1965), "Multi-Valued Monotone Nonlinear Mappings and Duality Mappings in Banach Spaces". *Transactions of the American Mathematical Society* **118**:338-351.

BROWDER, F. E. (1968), "Nonlinear Maximal Monotone Operators in Banach Space". *Mathematische Annalen* **175**:89-113.

BROWDER, F. E., PETRYSHN, W. V (1967), "Construction of Fixed Points of Nonlinear Mappings in Hilbert Spaces". *Journal of Mathematical Analysis and Applications* **20**:197-228.

BRUCK, R. E. (1973), "The Iterative Solution of the Equation $y \in x + Tx$ for a Monotone Operator T in Hilbert Space". *Bulletin of the American Mathematical Society* **79**(6):1258-1261.

BRUCK, R. E. (1974), "A Strongly Convergent Iterative Solution of $0 \in U(x)$ for a Maximal Monotone Operator U in Hilbert Space". *Journal of Mathematical Analysis and Applications* **48**:114-126.

BRUCK, R. E. (1975A), "An Iterative Solution of a Variational Inequality for Certain Monotone Operators in Hilbert Space". *Bulletin of the American Mathematical Society* **81**(5): 709-710, Corrigendum **82**(2):353.

BRUCK, R. E. (1975B), "Asymptotic Convergence of Nonlinear Contraction Semigroups in Hilbert Space". *Journal of Functional Analysis* **18**:15-26.

BRUCK, R. E. (1977), "On the Weak Convergence of an Ergodic Iteration for the Solution of Variational Inequalities in Hilbert Space". *Journal of Mathematical Analysis and Applications* **61**:159-164.

CESARI, L. (1983), *Optimization — Theory and Applications: Problems with Ordinary Differential Equations* (New York: Springer-Verlag).

COHEN, G., ZHU, D. L. (1984), "Decomposition Coordination Methods in Large Scale Optimization Problems: The Nondifferentiable Case and the Use of Augmented Lagrangians". In *Advances in Large-Scale Systems, Volume 1*, J. B. CRUZ, JR., editor (Greenwich: JAI Press).

COOK, W., GERARDS, A. M. H., SCHRIJVER, A., TARDOS, É. (1986), "Sensitivity Theorems in Integer Linear Programming". *Mathematical Programming* **34**:251-264.

DAFERMOS, S. (1980), "Traffic Equilibrium and Variational Inequalities". *Transportation Science* **14**:42-54.

DAFERMOS, S. (1983), "An Iterative Scheme for Variational Inequalities". *Mathematical Programming* **26**:40-47.

DANTZIG, G. B. (1963), *Linear Programming and Extensions* (Princeton: Princeton University Press).

DEBRUNNER, H., FLOR, P. (1964). "Ein Erweiterungssatz für monotone Mengen". *Archiv der Mathematik* **15**:445-447 (in German).

DEIMLING, K. (1985), *Nonlinear Functional Analysis* (Berlin: Springer-Verlag).

DOLEŽAL, V. (1979), *Monotone Operators and Applications in Control and Network Theory* (Amsterdam: Elsevier Scientific).

DOUGLAS, J., GUNN, J. E. (1964), "A General Formulation of Alternating Direction Methods. Part I. Parabolic and Hyperbolic Functions". *Numerische Mathematik* **6**:428-453.

DOUGLAS, J., RACHFORD, H. H. (1956), "On the Numerical Solution of Heat Conduction Problems in Two and Three Space Variables". *Transactions of the American Mathematical Society* **82**:421-439.

DUGUNDJI, J. (1966), *Topology*. (Boston: Allyn and Bacon)

DURIER, R. (1986) *On Locally Polyhedral Convex Functions*. Working paper, Université de Dijon.

DURIER, R., MICHELOT, C. (1986), "Sets of Efficient Points in a Normed Space". *Journal of Mathematical Analysis and its Applications* **117**(2):506-528.

ECKSTEIN, J. (1988), *The Lions-Mercier Algorithm and the Alternating Direction Method are Instances of the Proximal Point Algorithm.* Report LIDS-P-1769, Laboratory for Information and Decision Sciences, MIT.

EKELAND, I., TEMAM, R. (1976), *Convex Analysis and Variational Problems* (Amsterdam: North-Holland)

EREMIN, I. I. (1965), "A Generalization of the Motzkin-Agmon Relaxation Method". *Uspekhi Matemacheskaya Nauk* **22**(2):833-187 (in Russian).

EREMIN, I. I. (1966), "On Systems of Inequalities with Convex Functions in the Left Sides". *American Mathematical Society Translations* **88**:265-278.

ERMOL'EV, Y. M. (1966), "Methods for Solving Nonlinear Extremal Problems". *Kibernetika* **4**:1-17 (in Russian).

FAN, K. (1952) "Fixed-Point and Minimax Theorems in Locally Convex Topological Linear Spaces". *Proceedings of the National Academy of Sciences, USA* **38**:121-126.

FERZIGER, J. H. (1981), *Numerical Methods for Engineering Application* (New York: John Wiley).

FORTIN, M., GLOWINSKI, R. (1983), "On Decomposition-Coordination Methods Using an Augmented Lagrangian". In M. FORTIN, R. GLOWINSKI, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems* (Amsterdam: North-Holland).

GABAY, D. (1983), "Applications of the Method of Multipliers to Variational Inequalities". In M. FORTIN, R. GLOWINSKI, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems* (Amsterdam: North-Holland).

GABAY, D., MERCIER B. (1976), "A Dual Algorithm for the Solution of Nonlinear Variational Problems via Finite Element Approximations". *Computers and Mathematics with Applications* **2**:17-40.

GARCIA, C., ZANGWILL, W. (1981), *Pathways to Solutions, Fixed Points, and Equilibria* (Englewood Cliffs: Prentice-Hall).

GERMANOV, M. A., SPIRIDONOV, V. S. (1966), "On a Method of Solving Systems of Nonlinear Inequalities". *U.S.S.R. Computational Mathematics and Mathematical Physics* **6**(2):194-196.

GLICKSBERG, I. L. (1952), "A Further Generalization of the Kakutani Fixed Point Theorem, with Application to Nash Equilibrium Points". *Proceedings of the American Mathematical Society* **3**:170-174.

GLOWINSKI, R., LE TALLEC, P. (1987), *Augmented Lagrangian Methods for the Solution of Variational Problems*. MRC Technical Summary Report #2965, Mathematics Research Center, University of Wisconsin – Madison.

GLOWINSKI, R., MARROCO, A. (1975), "Sur L'Approximation, par Elements Finis d'Ordre Un, et la Resolution, par Penalisation-Dualité, d'une Classe de Problemes de Dirichlet non Lineares". *Revue Française d'Automatique, Informatique et Recherche Opérationelle* **9**(R-2):41-76 (in French).

GOLDBERG, A. V. (1987), *Efficient Graph Algorithms for Sequential and Parallel Computers*. Technical Report TR-374, Laboratory for Computer Science, MIT.

GOLDBERG, A. V., TARJAN, R. E. (1987), *Solving Minimum Cost Flow Problems by Successive Approximation*, Proceedings of the 19th ACM Symposium on the Theory of Computation.

GOLDSTEIN, A. A. (1964), "Convex Programming in Hilbert Space". *Bulletin of the American Mathematical Society* **70**: 709-710.

GOL'SHTEIN, E. G. (1975), "Method for Modification of Monotone Mappings". *Ekonomika i Matemacheskaya Metody* **11**(6):1142-1159 (in Russian).

GOL'SHTEIN, E. G. (1985), "Decomposition Methods for Linear and Convex Programming Problems". *Matekon* **22**(4):75-101.

GOL'SHTEIN, E. G. (1986), "The Block Method of Convex Programming". *Soviet Mathematics Doklady* **33**(3):584-587.

GOL'SHTEIN, E. G. (1987), "A General Approach to Decomposition of Optimization Systems". *Soviet Journal of Computer and Systems Sciences* **25**(3):105-114.

GOL'SHTEIN, E. G., TRET'YAKOV, N. V. (1975), "The Gradient Method of Minimization and Algorithms of Convex Programming Based on Modified Lagrangians Functions". *Ekonomika i Matemacheskaya Metody* **11**(4):730-742 (in Russian).

GOL'SHTEIN, E. G., TRET'YAKOV, N. V. (1979), "Modified Lagrangians in Convex Programming and their Generalizations". *Mathematical Programming Study* **10**:86-97.

GUBIN, L. G., POLYAK, B. T., RAIK, E. V. (1967), "The Method of Projections for Finding the Common Point of Convex Sets". *U.S.S.R. Computational Mathematics and Mathematical Physics* **7**(6):1-24.

HA, C.-D. (1980), *Decomposition Methods for Structured Convex Programming*. Doctoral dissertation, department of Industrial Engineering, University of Wisconsin — Madison.

HAN, S.-P., LOU, G. (1988), "A Parallel Algorithm for a Class of Convex Programs". *SIAM Journal on Control and Optimization* **26**(2):345-355.

HESTENES, M. R. (1969), "Multiplier and Gradient Methods". *Journal of Optimization Theory and Applications* **4**:303-320.

HOFFMAN, A. J. (1952), "On Approximate Solutions of Systems of Linear Inequalities". *Journal of Research of the National Bureau of Standards* **49**(4):263-265.

JOSHI, M. C., BOSE, R. K. (1985), *Some Topics in Nonlinear Functional Analysis* (New Dehli: Halsted Press/John Wiley).

KACHUROVSKII, R. I. (1960), "On Monotone Operators and Convex Functionals". *Uspekhi Matemacheskaya Nauk* **15**(4):213-215 (in Russian).

KACHUROVSKII, R. I. (1968), "Nonlinear Monotone Operators in Banach Space". *Russian Mathematical Surveys* **23**(2):117-165.

KACMARZ, M. S. (1937), "Angenäherte Auflösung von Systemen linearer Gleichungen". *Académie Polonaise des Sciences at des Lettres: Bulletin International Serie A* 355-357 (in German).

KAKUTANI, S. (1941), "A Generalization of Brouwer's Fixed Point Theorem". *Duke Mathematical Journal* **8**:457-459.

KELLOGG, R. B. (1969), "A Nonlinear Alternating Direction Method". *Mathematics of Computation* **23**:23-27.

KENDEROV, P. S. (1974), "The Set-Valued Monotone Mappings are Almost Everywhere Single-Valued". *Bulgarskaia Akademiia na Naukite, Sofia. Doklady* **27**(9)1173-1175.

KLINGMAN, D., NAPIER, A., STUTZ, J., (1974), "NETGEN — A Program for Generating Large-Scale (Un)capacitated Assignment, Transportation, and Minimum Cost Network Problems". *Management Science* **20**:814-822.

KORPELEVICH, G. M. (1976), "The Extragradient Method for Finding Saddle Points and Other Problems". *Matekon* **13**(4):35-49.

KORT, B., BERTSEKAS, D. (1972), "A New Penalty Function Method for Constrained Minimization". *Proceedings of the IEEE Conference on Decision and Control*, New Orleans, LA, 162-166.

KORT, B., BERTSEKAS, D. (1973), "Multiplier Methods for Convex Programming". *Proceedings of the IEEE Conference on Decision and Control*, San Diego, CA, 428-432.

KORT, B., BERTSEKAS, D. (1976), "Combined Primal-Dual and Penalty Methods for Convex Programming". *SIAM Journal on Control and Optimization* **14**(2):268-294.

LEFEBVRE O., MICHELOT, C. (1988), "About the Finite Convergence of the Proximal Point Algorithm". *Trends in Mathematical Optimization: 4th French-German Conference on Optimization*, International Series of Numerical Mathematics **84**, K.-H. HOFFMANN *ET. AL.*, editors (Basel: Birkhäuser-Verlag).

LEVITIN, E. S., POLYAK, B. T. (1966), "Constrained Minimization Methods". *U.S.S.R. Computational Mathematics and Mathematical Physics* **6**(5):1-50.

LIN, Y., CRYER, C. W. (1985), "An Alternating Direction Implicit Algorithm for the Solution of Linear Complementarity Problems Arising from Free Boundary Problems". *Applied Mathematics and Optimization* **13**(1):1-17.

LIONS, P.-L. (1978), "Une Méthode Itérative de Resolution d'une Inequation Variationnelle". *Israel Journal of Mathematics* **31**(2):204-208 (in French).

LIONS, P.-L., MERCIER, B. (1979), "Splitting Algorithms for the Sum of Two Nonlinear Operators". *SIAM Journal on Numerical Analysis* **16**(6):964-979.

LUENBERGER, D. G. (1973), *Linear and Nonlinear Programming* (Reading: Addison-Wesley).

LUQUE, F. J. (1984A), "Asymptotic Convergence Analysis of the Proximal Point Algorithm". *SIAM Journal on Control and Optimization* **22**(2):277-293.

LUQUE, F. J. (1984B), *Nonlinear Proximal Point Algorithms*. Doctoral Thesis, MIT.

LUQUE, F. J. (1986A), *The Nonlinear Proximal Point Algorithm and Multiplier Methods*. Report LIDS-P-1596, Laboratory for Information and Decision Sciences, MIT.

LUQUE, F. J. (1986B), *Convolutions of Maximal Monotone Mappings*. Report LIDS-P-1597, Laboratory for Information and Decision Sciences, MIT.

LUQUE, F. J. (1986C), *The Nonlinear Proximal Point Algorithm*. Report LIDS-P-1598, Laboratory for Information and Decision Sciences, MIT.

MAGNANTI, T. L. (1974), "Fenchel and Lagrange Duality are Equivalent". *Mathematical Programming* **7**(2):253-258.

MANGASARIAN, O. L. (1981), "A Condition Number for Linear Inequalities and Linear Programs". *Methods of Operations Research* **43**:3-15 (Proceedings of the 6. Symposium über Operations Research, Augsburg, September 1981, G. BAMBERG and O. OPITZ, editors. Cambridge: Oelgeschlager, Gunn, and Hain.)

MANGASARIAN, O. L., SHIAU T.-H. (1987), "Lipschitz Continuity of Solutions of Linear Inequalities, Programs and Complementarity Problems". *SIAM Journal on Control and Optimization* **25**(3):583-595.

MARCHUK, G. I. (1975), *Methods of Numerical Mathematics* (New York: Springer-Verlag).

MARTINET, B. (1970), "Regularisation d'Inequations Variationelles par Approximations Successives". *Revue Française d'Informatique et de Recherche Operationelle* **4**(R-3):154-158 (in French).

MARTINET, B. (1972), "Determination Approchée d'un Point Fixe d'une Application Pseudo-Contractante. Cas de l'Application prox". *Comptes Rendus de l'Academie des Sciences, Paris, Série A* **274**:163-165 (in French).

MINTY, G. J. (1961), "On the Maximal Domain of a 'Monotone' Function". *Michigan Mathematical Journal* **8**:135-137.

MINTY, G. J. (1962), "Monotone (Nonlinear) Operators in Hilbert Space". *Duke Mathematics Journal* **29**:341-346.

MINTY, G. J. (1964), "On the Monotonicity of the Gradient of a Convex Function". *Pacific Journal of Mathematics* **14**:243-247.

MOREAU, J.-J. (1962), "Fonctions Convexes Duales at Points Proximaux dans un Espace Hilbertien". *Comptes Rendus de l'Academie des Sciences, Paris* **255**:2897-2899 (in French).

MOREAU, J.-J. (1965), "Proximité et Dualité dans un Espace Hilbertien". *Bulletin de la Societé Mathematique de France* **93**:273-299 (in French).

MOTZKIN, T. S., SCHOENBERG, I. J. (1954), "The Relaxation Method for Linear Inequalities". *Canadian Journal of Mathematics* **6**:393-404.

OPIAL, Z. (1967), "Weak Convergence of the Sequence of Successive Approximations for Nonexpansive Mappings". *Bulletin of the American Mathematical Society* **73**:591-597.

PAPADIMITRIOU, C. H., STEIGLITZ, K. (1982), *Combinatorial Optimization: Algorithms and Complexity* (Englewood Cliffs: Prentice-Hall).

PASCALI, D., SBURLAN, S. (1978), *Nonlinear Mappings of Monotone Type* (Bucarest: Editura Academeie).

PASSTY, G. B. (1979), "Ergodic Convergence to a Zero of the Sum of Monotone Operators in Hilbert Space", *Journal of Mathematical Analysis and Applications* **72**:383-390.

PEACEMAN, D. W., RACHFORD, H. H. (1955), "The Numerical Solution of Parabolic and Elliptic Differential Equations". *SIAM Journal* **3**:28-41.

POLYAK, B. T. (1967), "A General Method for Solving Extremum Problems". *Soviet Mathematics Doklady* **8**(3):593-597.

POLYAK, B. T. (1969), "Minimization of Nonsmooth Functionals". *U.S.S.R. Computational Mathematics and Mathematical Physics* **9**(3):14-29.

POLYAK, B. T. (1970), "Iterative Methods Using Lagrange Multipliers for Solving Extremal Problems with Constraints of the Equation Type". *U.S.S.R. Computational Mathematics and Mathematical Physics* **10**(5):42-52.

POLYAK, B. T. (1987), *Introduction to Optimization* (New York: Optimization Software).

POWELL, M. J. D. (1969), "A Method for Nonlinear Constraints in Minimization Problems", in *Optimization*, R. FLETCHER, editor (New York: Academic Press).

REINERMAN, J., SCHÖNEBERG, R. (1976), "Some Results in Fixed Point Theory for Nonexpansive and Pseudocontractive Maps in Hilbert Space", in *Fixed Point Theory and its Applications*, S. SWAMINATHAN, editor (New York: Academic Press).

ROBINSON, S. M. (1977), "A Characterization of Stability in Linear Programming". *Operations Research* **25**(3):435-447.

ROCKAFELLAR, R. T. (1966), "Characterization of the Subdifferentials of Convex Functions". *Pacific Journal of Mathematics* **17**(3):497-510.

ROCKAFELLAR, R. T. (1969), "Local Boundedness of Nonlinear, Monotone Operators". *Michigan Mathematical Journal* **16**:397-407.

ROCKAFELLAR, R. T. (1970A), *Convex Analysis* (Princeton: Princeton University Press).

ROCKAFELLAR, R. T. (1970B), "On the Maximal Monotonicity of Subdifferential Mappings". *Pacific Journal of Mathematics* **33**(1):209-216.

ROCKAFELLAR, R. T. (1970C), "On the Maximality of Sums of Nonlinear Monotone Operators". *Transactions of the American Mathematical Society* **149**:75-88.

ROCKAFELLAR, R. T. (1970D), "On the Virtual Convexity of the Domain and Range of a Nonlinear Maximal Monotone Operator". *Mathematische Annalen* **185**:81-90.

ROCKAFELLAR, R. T. (1976A), "Monotone Operators and the Proximal Point Algorithm". *SIAM Journal on Control and Optimization* **14**(5):877-898.

ROCKAFELLAR, R. T. (1976B), "Augmented Lagrangians and Applications of the Proximal Point Algorithm in Convex Programming". *Mathematics of Operations Research* **1**(2):97-116.

ROCKAFELLAR, R. T. (1977), "Monotone Operators and and Augmented Lagrangian Methods in Nonlinear Programming", in *Nonlinear Programming 3*, O. L. MANGASARIAN, R. M. MEYER, S. M. ROBINSON, editors (New York: Academic Press).

ROCKAFELLAR, R. T. (1984), *Network Flows and Monotropic Optimization* (New York: John Wiley).

ROCKAFELLAR, R. T., WETS R. J.-B. (1987), *Scenarios and Policy Aggregation in Optimization under Uncertainty*. Working paper WP-87-119, International Institute for Applied Systems Analysis (IIASA), Laxenberg, Austria.

RUSZCZYNSKI, A. (1986), "A Regularized Decomposition Method for Minimizing a Sum of Polyhedral Functions". *Mathematical Programming* **35**:309-333.

RUSZCZYNSKI, A. (1988A), *Regularized Decomposition and Augmented Lagrangian Decomposition for Angular Linear Programming Problems*. Working paper WP-88-071, International Institute for Applied Systems Analysis (IIASA), Laxenberg, Austria.

RUSZCZYNSKI, A. (1988B), *Parallel Decomposition of Multistage Stochastic Programming Problems*. Working paper WP-88-094, International Institute for Applied Systems Analysis (IIASA), Laxenberg, Austria.

SCHRIJVER, A. (1986), *Theory of Linear and Integer Programming* (Chichester: John Wiley).

SHAPIRO, J. F. (1979), *Mathematical Programming: Structures and Algorithms* (New York: John Wiley).

SHOR, N. Z. (1970), "Utilization of the Operation of Space Dilation in the Minimization of Convex Functions". *Kybernetika* **6**:6-12 (in Russian).

SIBONY, M. (1970), "Méthodes Itératives pour les Équations et Inéquations aux Dérivees Partielles Non Linéaires de Type Monotone". *Calcolo* **7**:65-183.

SPINGARN, J. E. (1983), "Partial Inverse of a Monotone Operator". *Applied Mathematics and Optimization* **10**:247-265.

SPINGARN, J. E. (1985A), "A Primal-Dual Projection Method for Solving Systems of Linear Inequalities". *Linear Algebra and Its Applications* **65**:45-62.

SPINGARN, J. E. (1985B), "Application of the Method of Partial Inverses to Convex Programming: Decomposition". *Mathematical Programming* **32**:199-233.

STEPHANOPOULOS, G., WESTERBERG, A. W. (1975), "The Use of Hestenes' Method of Multipliers to Resolve Dual Gaps in Engineering Systems Optimization". *Journal of Optimization Theory and Applications* **15**(3):285-309.

STRANG, G. (1976), *Linear Algebra and its Applications* (Orlando: Academic Press).

TANK, D. W., HOPFIELD, J. J. (1986), "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit". *IEEE Transactions on Circuits and Systems* **33**(5):533-541.

TARDOS, E. (1985), "A Strongly Polynomial Minimum Cost Circulation Algorithm", *Combinatorica* **5**:247-255.

TSENG, P. (1988), *Applications of a Splitting Algorithm to Decomposition in Convex Programming and Variational Inequalities*. Report LIDS-P-1836, Laboratory for Information and Decision Sciences, MIT.

TSENG, P., BERTSEKAS, D. P., TSITSIKLIS, J. N. (1988), *Partially Asynchronous, Parallel Algorithms for Network Flow and Other Problems*. Report LIDS-P-1832, Laboratory for Information and Decision Sciences, MIT.

VAĬNBERG, M. M. (1973), *Variational Method and Method of Monotone Operators* (Jerusalem: Halsted Press/John Wiley).

VON NEUMAN, J. (1950), *Functional Operators Volume II: The Geometry of Orthogonal Spaces*, Annals of Mathematics Studies **22** (Princeton: Princeton University Press).

WATANABE, N., NISHIMURA, Y., MATSUBARA, M. (1978), "Decomposition in Large System Optimization Using the Method of Multipliers". *Journal of Optimization Theory and Applications* **25**(2):181-193.

WILLIAMS, A. C. (1963), "Marginal Values in Linear Programs". *SIAM Journal* **11**(1): 82-94.

ZEIDLER, E. (1985), *Nonlinear Functional Analysis and its Applications III: Variational Methods and Optimization* (New York: Springer-Verlag).

ZENIOS, S. A. (1988A), *Numerical Optimization Benchmarks on Advanced Architecture Computers*. Report 88-04-03, Decision Sciences Department, Wharton School, University of Pennsylvania.

ZENIOS, S. A. (1988B), *Matrix Balancing on a Massively Parallel Connection Machine*. Report 88-09-04, Decision Sciences Department, Wharton School, University of Pennsylvania.

ZENIOS, S. A. (1989), "Parallel Numerical Optimization: Current Status and an Annotated Bibliography". *ORSA Journal on Computing* **1**(1):20-43.

ZENIOS, S. A., LASKEN, R. A. (1988), "Nonlinear Network Optimization on a Massively Parallel Connection Machine". *Annals of Operations Research* **14**:147-165