Portfolio Selection Under Nonsmooth Convex Transaction Costs

by

Marina Potaptchik

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2006

I hereby declare that I am the sole author of this thesis. This is the true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We consider a portfolio selection problem in the presence of transaction costs. Transaction costs on each asset are assumed to be a convex function of the amount sold or bought. This function can be nondifferentiable in a finite number of points. The objective function of this problem is a sum of a convex twice differentiable function and a separable convex nondifferentiable function. We first consider the problem in the presence of linear constraints and later generalize the results to the case when the constraints are given by the convex piece-wise linear functions.

Due to the special structure, this problem can be replaced by an equivalent differentiable problem in a higher dimension. It's main drawback is efficiency since the higher dimensional problem is computationally expensive to solve.

We propose several alternative ways to solve this problem which do not require introducing new variables or constraints. We derive the optimality conditions for this problem using subdifferentials. First, we generalize an active set method to this class of problems. We solve the problem by considering a sequence of equality constrained subproblems, each subproblem having a twice differentiable objective function. Information gathered at each step is used to construct the subproblem for the next step. We also show how the nonsmoothness can be handled efficiently by using spline approximations. The problem is then solved using a primal-dual interior-point method.

If a higher accuracy is needed, we do a crossover to an active set method. Our numerical tests show that we can solve large scale problems efficiently and accurately.

Acknowledgements

It is my great pleasure to thank here my thesis advisor Professor Michael Best for his constant encouragement and support without which this work would have hardly been done. Mike has given me enormous freedom to pursue my own interests while at the same time providing just the right amount of guidance when I needed it.

I have benefited a lot from various discussions with Professor Phelim Boyle, to whom I also extend my sincerest thanks.

I also would like to thank the Department of Combinatorics and Optimization at the University of Waterloo for providing a stimulating and encouraging research environment.

Special thanks to Professors Levent Tunçel and Henry Wolkowicz for many enlightening research sessions.

I would like to thank Professors Michael Best, Katta Murty, Phelim Boyle, Henry Wolkowicz and Levent Tunçel for serving on my examining committee. Their comments and suggestions are appreciated.

I am grateful to the Natural Science and Engineering Research Council of Canada, the Government of Ontario, the University of Waterloo and my supervisor for their financial support.

My most sincere thanks to my husband Alex for his help and support.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Definition

We consider the problem of selecting a portfolio for an investor in an optimal way. There are many ways to define optimality for this model.

One of the earliest and most popular models is the mean-variance model proposed by Markowitz [41]. It is assumed that an investor wants to maximize the expected return of the portfolio while minimizing the variance. The balance between these two conflicting goals is chosen depending on the investor's risk tolerance.

Another, more general, approach to selecting an optimal portfolio is maximizing an expected utility of portfolio return. The utility function of an investor measures the satisfaction associated with a particular level of wealth. The investor's aversion to risk implies that the utility function is concave. This function is also assumed to be twice differentiable.

The mean variance analysis is consistent with the expected utility maximization under some assumptions.

Assume that n assets are available. We denote by $x = (x_1, x_2, \ldots, x_n)^T$ the vector of holdings in each asset. We denote by $R = (R_1, R_2, ..., R_n)^T$ the rates of return on these assets. Note that R is a random variable.

Under the mean-variance model, the investor acts to maximize the function

$$f(x) = t\mu^T x - \frac{1}{2} x^T \Sigma x,$$

where μ is the vector of the expected returns of the assets, Σ is a covariance matrix and t is a fixed scalar parameter.

Under the expected utility model, the investor acts to maximize the expectation of his utility function of the final wealth $U(w_0(1 + R^T x))$, where w_0 denotes the initial wealth of the investor. One commonly used utility function is the power utility function

$$U(w_0(1 + R^T x)) = -\frac{1}{\gamma}(w_0(1 + R^T x))^\gamma,$$

where $0 < \gamma < 1$.

See [30] for more details on the expected utility approach to portfolio optimization.

In practice, every time an investor buys or sells some of his holdings, he incurs a certain extra cost, called the transaction cost. Two major sources of transaction costs are brokerage fees and market impact costs.

The broker's commission rates are often decreasing in the amount of trade, and therefore the transaction costs resulting from these fees are modeled by concave functions. However, this is only the case when the amounts of transactions are not very high and should be taken into account by smaller investors. If the trade volume is large enough, the commissions can be modeled by a linear function.

The market impact costs are the changes in the price of the assets that result from large amounts of these assets being bought or sold: the price is going up if someone is buying large quantities of an asset, and the price is going down if a lot of shares of this asset are for sale. The market impact costs are normally modeled by convex functions. The piece-wise linear function is the most common example.

So, from the point of view of a large institutional investor, the transaction costs can be adequately modeled by a convex function.

We assume that vector $\hat{x} = (\hat{x_1}, \hat{x_2}, \ldots, \hat{x_n})^T$ represents the current holdings of assets. The cost associated with changing the holdings in asset i from $\hat{x}_i$ to x_i will be denoted by $f_i(x_i)$.

For most practical purposes, it is safe to assume that transaction costs on each asset depend only on the amount of the holdings in this asset purchased or sold and do not depend on the amount of transactions in other assets. Therefore, we model the transaction costs by a separable function of the amount sold or bought, i.e. the cost associated with changing the portfolio from $\hat{x}$ to x is $f(x) = \sum_{i=1}^{n} f_i(x_i)$.

Let us consider a simple example. Suppose only 2 assets are available ($n = 2$) and suppose that at the present moment we have all the money invested in asset number 1. Our current holdings are represented by vector $\hat{x} = (1, 0)^T$. Let the transaction costs be equal to 1% of the amount bought or sold for each asset:

$$f_i(x_i) = 0.01|x_i - \hat{x_i}| = \begin{cases} 0.01(x_i - \hat{x_i}), & \text{if} \quad x_i \geq \hat{x_i}, \\ 0.01(\hat{x_i} - x_i), & \text{if} \quad x_i < \hat{x_i}, \end{cases}$$

for $i = 1, 2$.

Note that the functions $f_1(x_1)$ and $f_2(x_2)$ are nondifferentiable at $\hat{x}_1$ and $\hat{x}_2$, respectively. (See Figure 1.1.)

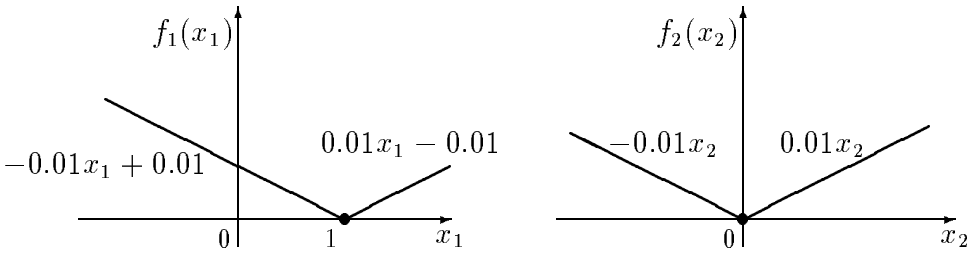

Figure 1.1: Transaction costs functions $f_1(x_1)$ and $f_2(x_2)$.

In what follows, we assume the following model for the transaction costs. The costs are given by $f^+_{i0}(x_i - \hat{x}_i)$ if an investor is buying up to d^+_{i1} units of asset i. If the investor needs more than d^+_{i1} but less than d^+_{i2}, he pays $f^+_{i0}(d^+_{i1})$ in transaction costs for the first d^+_{i1} units and $f^+_{i1}(x_i - \hat{x}_i - d^+_{i1})$ for the amount that exceeds d^+_{i1}. If an investor needs more than d^+_{i2} units of the i-th asset, he has to pay a different rate for the amount exceeding d^+_{i2}, etc. Suppose M^+_i is the number of intervals corresponding to the different transaction rates for buying asset i. On the portion of the purchase exceeding $d^+_{iM^+_i}$ the transaction costs are equal to $f^+_{iM^+_i}(x_i - \hat{x}_i - d^+_{iM^+_i})$. If an investor sells some amount of asset i, transaction costs have similar structure. The investor pays $f^-_{i0}(d^+_{i1})$ in transaction costs for the first d^+_{i1} units, $f^-_{i1}(-x_i + \hat{x}_i - d^+_{i1})$ for the amount that exceeds d^+_{i1} but less than d^-_{i2}, etc. The number of intervals corresponding to the different transaction rates for selling asset i is denoted by M^-_i.

If the holding in the asset number i has not changed, i.e. $x_i = \hat{x}_i$, the transaction costs associated with this asset should be equal to zero. Therefore $f_i(x)$ should satisfy the conditions

$$f_i(\hat{x}_i) = 0. \tag{1.1}$$

For consistency of notation we also set $d^+_{i0} = d^-_{i0} = 0$, $d^+_{iM^+_i+1} = d^-_{iM^-_i+1} = +\infty$.

More formally, the costs associated with changing the holdings in asset i from $\hat{x}_i$ to x_i are given by the following function

$$f_i(x_i) = \begin{cases} f_{il}^-(-x_i + \hat{x}_i - d_{il}^-) + \sum_{j=1}^{l} f_{ij-1}^-(d_{ij}^-), \text{ if } x_i - \hat{x}_i \in [-d_{il+1}^-, -d_{il}^-], \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for some } l \in \{0, .., M_i^-\}, \\ f_{il}^+(x_i - \hat{x}_i - d_{il}^+) + \sum_{j=1}^{l} f_{ij-1}^+(d_{ij}^+), \quad \text{if } x_i - \hat{x}_i \in [d_{il}^+, d_{il+1}^+], \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{for some } l \in \{0, .., M_i^+\}. \end{cases}$$

The above notation comes naturally from the statement of the problem, but we can simplify it for the purpose of formulating the solution algorithm. Let $M_i = M_i^+ + M_i^- + 1$, so that M_i is the total number of end points of the intervals ("break points"). We further denote

$$\begin{aligned} d_{il} &= \hat{x}_i - d_{i(M_i^- - l + 1)}^-, \quad l = 0, ..., M_i^- + 1, \\ d_{il} &= \hat{x}_i + d_{i(l - M_i^+ + 1)}^+, \quad l = M_i^- + 2, ..., M_i + 1, \end{aligned}$$

and

$$\begin{aligned} f_{il}(x_i) &= f_{i(M_i^- - l)}^-(-x_i + \hat{x}_i - d_{i(M_i^- - l)}^-) + \sum_{j=1}^{(M_i^- - l)} f_{ij-1}^-(d_{ij}^-), \; l = 0, ..., M_i^-, \\ f_{il}(x_i) &= f_{i(l - M_i^+)}^+(x_i - \hat{x}_i - d_{i(l - M_i^+)}^+) + \sum_{j=1}^{(l - M_i^+)} f_{ij-1}^+(d_{ij}^+), \quad l = M_i^- + 1, ..., M_i. \end{aligned}$$

Thus we can rewrite the cost functions in the following more compact way:

$$f_i(x_i) = \begin{cases} f_{i0}(x_i), & \text{if} \quad x_i \le d_{i1}, \\ f_{il}(x_i), & \text{if} \quad x_i \in [d_{il}, d_{il+1}], \quad l = 1, .., M_i, \\ f_{iM_i}(x_i), & \text{if} \quad x_i \ge d_{iM_i}. \end{cases} \tag{1.2}$$

We assume that f_{il} are twice differentiable on (d_{il}, d_{il+1}), f_{i0} are twice differentiable on $(-\infty, d_{i1})$ and f_{iM_i} are twice differentiable on $(d_{iM_i}, +\infty)$ for each $i = 1, .., n$, $l = 0, .., M_i$, and in addition that (1.1) are satisfied. Figure 1.2 gives an example of such a function.

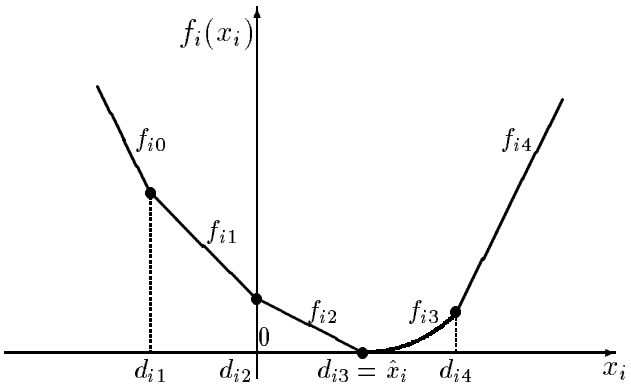

Figure 1.2: An example of a transaction cost function.

The investor acts to maximize an expected utility function of his holdings in these assets, while taking into account the transaction costs. Note that maximizing a utility function is equivalent to minimizing the negative of it. So we will model the objective of maximizing the expected utility of the investor in presence of transaction costs by

$$\begin{array}{ll} \min & F(x) + \sum_{i=1}^{n} f_i(x_i), \\ \text{s.t.} & Ax \leq b, \end{array} \tag{1.3}$$

where $F(x)$ is the negative utility function and $Ax \leq b$ is a set of linear constraints.

1.2 Overview

Robust Optimization

In this thesis we concentrate on the linear and piece-wise liner constraints. This is not always the case in practice. Constraints like bounds on the variance of the return, and bounds on different shortfall probabilities require a different approach, typically these constraints are modeled by quadratic functions.

Another important generalizations of the portfolio optimization problem is accounting for the uncertainty in the data. If the parameters of the problem are not specified exactly but they are only known to belong to a given uncertainty set, such optimization problem is called a Robust Optimization Problem.

Consider, for example, the robust counterpart of the linearly constrained mean-variance portfolio problem

$$\begin{array}{ll} \min & t\mu^T x - \frac{1}{2} x^T \Sigma x, \\ \text{s.t.} & a_i^T x \leq b_i,\ i = 1, \ldots, m, \\ & \mu \in \mathcal{M},\ \Sigma \in \mathcal{S},\ a_i \in \mathcal{A}_i,\ b \in \mathcal{B}. \end{array} \tag{1.4}$$

where $\mathcal{M},\ \mathcal{A}_i \subset \mathbb{R}^n,\ \mathcal{B} \subset \mathbb{R}^m$ and $\mathcal{S} \subset \mathbb{R}^{n \times n}$ define the range of uncertainty for the estimates of the data. These sets are typically represented by box constraints, ellipsoidal or $\cap$-ellipsoidal constraints. Under this assumption, the robust problem (1.4) can be solved efficiently. For more details on the robust optimization and robust portfolio problems see [3], [25], [37], [38].

In the case when problem (1.4) has a constant covariance matrix Σ (and possibly additional quadratic constraints), this problem can be represented by a Second-Order Cone Programming Problem (SOCP)

$$\begin{array}{ll} \min & \bar{c}^T y, \\ \text{s.t.} & \|\bar{A}_i y\ + \bar{b}_i\| \leq \bar{d}_i^T y\ + \bar{e}_i,\ i = 1, \ldots, M. \end{array} \tag{1.5}$$

where $\|\ \|$ denotes the Euclidean norm.

If covariance matrix $\Sigma \in \mathcal{S}$ is not a constant matrix, it is critical that every element of the set $\mathcal{S}$ must be positive semidefinite, since they must be valid covariance matrices. In this case problem (1.4) becomes a Semi-definite Program (SDP).

Both SDPs and SOCPs have been studied extencivelly in recent years and efficient interior-point algorithms are available for solving these problems. See [43],

[56], [44].

Nondifferentiable Optimization

A large number of methods are available for solving nondifferentiable optimization problems. In this section, we give a brief overview of the most important classes of methods. For a more detailed review see Lemaréchal [35], Bertsekas [4], or the book by Kiwiel [32].

The subgradient method was introduced and extensively developed by Shor, see [50], [51]. This method generates a sequence of feasible points, using a formula similar to the one of the gradient or gradient projection method, except that a subgradient is used in place of gradient. Unlike in the differentiable case, most methods compute only one element of the subdifferential at the point of iteration. For more details in the convex case, see Hiriart-Urruty and Lemaréchal [29], while for the locally Lipschitz case a good overview is Kiwiel [32]. Burke et al. [13] provide an interesting approach via the random sampling of gradients at nearbypoints. For a survey on different ways of generalizing the notion of derivative to the nonsmooth case and it's use in optimization see Dutta [20].

The cutting plane method is due independently to Cheney and Goldstein [14] and Kelley [31]. At each iteration of this method, the objective function of the problem is replaced by a polyhedral approximation, constructed using the points generated so far and their subgradients. One of the drawbacks of this method is that it can take large steps away from the optimum even when a good initial point is given.

A more sophisticated versions of this method are proximal cutting plane method or bundle method. The main idea here is to add to the polyhedral objective function a quadratic penalty term, which makes the method more stable. For analysis of this

method see Lemaréchal [35], Lemaréchal, Strodiot, Bihain,[36] and Makela [40].

The central cutting plane method considers the same polyhedral approximation as the cutting plane method, but instead of finding a minimum of this function, it generates a so-called "central pair". This method is related to the interior point methods and benefits from advances in this area. The method was introduced by Elzinga and Moore [21]. For a survey see Goffin and Vial [26].

All the methods mentioned above are designed to handle a wide set of nondifferentiable problems. In some problems, the nondifferentiability has a structure that makes them amenable to a differentiable method, see Lemaréchal [35] for such examples. Our problem falls into this category.

Because of the special structure of the nondifferentiable part of the objective function, problem (1.3) can be converted to a smooth one by introducing new variables and constraints into the problem. For example, for each $i = 1, \ldots, n$, we can introduce a new set of variables x_{il}^+ where $l = 0, \ldots, M_i^+$ and x_{il}^- where $l = 0, \ldots, M_i^-$ .Then problem (1.3) can be rewritten in the form

$$\begin{aligned} \min \quad & F(x) + \textstyle\sum_{i=1}^{n} \sum_{l=0}^{M_i^+} f_{il}^+(x_{il}^+) + \sum_{i=1}^{n} \sum_{l=0}^{M_i^-} f_{il}^-(x_{il}^-) \\ s.t. \quad & Ax \leq b, \\ & x_i - \textstyle\sum_{l=0}^{M_i^+} x_{il}^+ + \sum_{l=0}^{M_i^-} x_{il}^- = \hat{x}_i, \ \text{ for } i = 0, ..., n, \\ & 0 \leq x_{il}^+ \leq d_{il+1}^+, \ \text{ for } i = 1, ..., n, \ l = 0, ..., M_i^+, \\ & 0 \leq x_{il}^- \leq d_{il+1}^-, \ \text{ for } i = 1, ..., n, \ l = 0, ..., M_i^-. \end{aligned} \tag{1.6}$$

The proof of the equivalence is given in Lemma A.1.

The problem (1.6) is a linearly constrained, convex and twice differentiable problem. Active set methods or interior point methods can be used to solve it. This is the most straightforward and traditional approach to solving this problem. It's most severe drawback is that the higher dimensional problem is computationally

expensive to solve.

In some cases one can develop an algorithm which is specially tailored for solving (1.3) and does not require introducing new variables or constraints. A dual active set algorithm for a special case of (1.3) is proposed in [33]. The very special structure of the constraints in this case allowed to reformulate the dual in convenient way. In [49], a simplex-type algorithm is proposed for solving (1.3) for the case when objective function is separable and quadratic.

A special case of (1.6), when each of the functions $f_i(x)$ may not be differentiable at only one point $\hat{x}_i$, was considered in [6]. The authors showed that it is possible to eliminate the newly introduced variables from the optimality conditions and derived a solution algorithm in terms of n-dimensional quantities.

Linear Equality Constrained Optimization

Consider the equality constrained problem

$$\begin{aligned} \min \quad & \theta(x) \\ \text{s. t.} \quad & Ax = b. \end{aligned} \tag{1.7}$$

where A is a matrix of order $m \times n$ and rank m, and θ is a continuously differentiable function. Subproblems of this type are important component of the algorithm proposed in this thesis.

The first order necessary optimality conditions for this problem are

$$\begin{aligned} & \nabla(\theta(x)) = 0, \\ & Ax = b. \end{aligned} \tag{1.8}$$

If the objective function is convex, these conditions are also sufficient for optimality. If x_0 is feasible for (1.7), any feasible solution for this problem has a form $x_0 + s$, where $As = 0$. We can rearrange the columns of A so that A and s can be

partitioned into a basic and non-basic parts as $[A_b, A_n]$ and $[s_b, s_n]$, where A_b is a square non-singular matrix of order m. Then $s_b = -A_b^{-1} A_n s_n$ and we can introduce a matrix S

$$S = \begin{bmatrix} -A_b^{-1} A_n \\ I_{n-m} \end{bmatrix}.$$

Note that the problem (1.7) is equivalent to minimizing $f(s_n) = \theta(x_0 + S s_n)$ over $s_n \in \mathbb{R}^{n-m}$ and the methods of unconstrained minimization can be applied to it. The feasible search direction s is called a descent direction at x_0 if

$$\nabla f(x_0)^T s_n = ((\nabla \theta(x_0))S)^T s_n = (\nabla \theta(x_0))^T s < 0.$$

If the steepest descent method is used, the direction

$$s = -SS^T (\nabla \theta(x_0))^T$$

defines the **projected gradient method**.

If the function θ is twice continuously differentiable, we denote by $H(\theta(x))$ the Hessian of $\theta(x)$. Then Newton's method can be applied to the unconstrained problem, the direction $s = S s_n$ is found from

$$(S^T (H\theta(x_0))S) s_n = -S^T (\nabla \theta(x_0))^T.$$

This defines the **projected Hessian method**. Quasi-Newton methods can also be developed for this problem.

Active Set Methods

The method that we are proposing for solving (1.3) is closely related to the active set method. This is a feasible direction method for the linearly constrained problem

$$\begin{aligned} &\min \quad \theta(x) \\ &\text{s. t.} \quad a_i x \le b_i, \ i = 1, \ldots, m, \end{aligned} \tag{1.9}$$

where $x \in \mathbb{R}^n$ and $\theta(x)$ is a real valued continuously differentiable function. If x is feasible for (1.9), the i-th constraint is said to be active at x if it holds as an equality, i.e. $a_i x = b_i$. Let us denote by $I(x)$ the index set of the constraints active at a feasible point x:

$$I(x) = \{i \mid a_i x = b_i\}.$$

If $\bar{x}$ is optimal for (1.9) then $\bar{x}$ is also optimal for the equality constrained problem

$$\begin{aligned} \min \quad & \theta(x) \\ \text{s. t.} \quad & a_i x = b_i, \ i \in I(\bar{x}). \end{aligned} \tag{1.10}$$

If we knew the active set corresponding to the optimal solution $\bar{x}$ of (1.9), we could solve the equality constrained problem (1.10) using one of the methods for the equality constrained optimization discussed above, and $\bar{x}$ would be among the solutions of (1.10).

In the active set method, a guess is built up over the steps, on the likely set of active constraint indices at the optimum. This set is known as a working active set. The working active set in the step $j+1$ is denoted by I^j. The method is started with a feasible point x_0.

Given an iterate x^j, we try to find a search direction s^j such that

$$\theta(x^j + s^j) < \theta(x^j) \text{ and } a_i s^j = 0, \ i \in I^j. \tag{1.11}$$

We have two possibilities:

(a) Vector s^j satisfying (1.11) is found. Then the next point is given by

$$x^{j+1} = x^j + \sigma^j s^j,$$

where the step size σ^j is chosen to guarantee the feasibility of x^{j+1} and sufficient decrease in the objective function. New indices might be added to the working active set at this step.

(b) Vector x^j is stationary for the problem (1.10). If the first order optimality conditions for (1.9) are also satisfied by x^j, algorithm terminates with x^j.

Otherwise, one of the indices is deleted from the working active set. Under the standard nondegeneracy assumptions, the new subproblem is guaranteed to have a solution with a strictly smaller objective function value.

For more details see [42].

In the case when the objective function is convex quadratic this algorithm terminates with an optimal solution after a finite number of iterations. The algorithm generally performs best when a good initial point is available.

Interior Point Methods

The main idea of the interior point methods is to add to the objective function $\theta(x)$ a barrier function $B(x)$. The latter should be continuous on the interior of the feasible set and should go to infinity if at least one of the inequality constraints becomes active.

The method is defined by introducing a parameter sequence $\{\mu^k\}$ with

$$0 < \mu^{k+1} < \mu^k, \ k = 0, 1, \ldots, \ \mu^k \to 0.$$

At the k-th iteration

$$x^k = \arg\min\{\theta(x) - \mu^k B(x)\}, \ k = 0, 1, \ldots.$$

It can be shown that it is sufficient to solve each subproblem approximately. This fact makes the interior point methods much more efficient. For more details on this approach see [24].

An alternative, increasingly popular approach, is to use primal-dual method, where primal variables x dual variable u and slack variables s are treated as independent variables. The optimality conditions for (1.3) are modified as follows

$$\begin{aligned} -\nabla\theta(x) &= A^T u,\ u \geq 0, \\ Ax + s &= b,\ s \geq 0, \\ s_i u_i &= \mu,\ i = 1, \ldots, m, \end{aligned} \tag{1.12}$$

and Newton's method is applied to system (1.12). Thus, given a current iterate (x, u, s), satisfying $(u, s) > 0$, the primal-dual steps $(\Delta x, \Delta u, \Delta s)$ are obtained from

$$\begin{bmatrix} H(\theta(x)) & A^T & 0 \\ A & 0 & I \\ 0 & S & U \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta u \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -Us + \sigma\mu e \end{bmatrix}, \tag{1.13}$$

where

$$r_c = \nabla\theta(x) + H(\theta(x)) + A'u,$$

$$r_b = Ax + s - b,$$

$$\mu = \frac{u^T s}{m},$$

$$U = \mathrm{diag}(u_1, \ldots, u_m),\ S = \mathrm{diag}(s_1, \ldots, s_m).$$

$$e = (1, \ldots, 1)^T,\ \sigma \in [0, 1].$$

The above system can be rewritten in a more compact way

$$[G + Q + A^T(S^{-1}U)A]\Delta x = -r_c + A^T(S^{-1}U)[-r_b + s - \sigma\mu U^{-1}e], \tag{1.14}$$

using

$$\Delta u = S^{-1}U[A\Delta x + r_b - s + \sigma\mu U^{-1}e]$$

and

$$\Delta s = -U^{-1}S\Delta u - s + \sigma\mu U^{-1}e.$$

1.3 Outline of the Thesis

The main contribution of the thesis is two efficient algorithms for solving problem (1.3).

In Chapter 2 we derive the optimality conditions for (1.3) using subdifferentials. This allows us to develop a computational algorithm (Algorithm 1) which is a generalization of the active set method. This method solves the problem by considering a sequence of differentiable equality constrained subproblems. Information gathered at each iteration is used to construct the next one.

In Chapter 3 we generalize the results of Chapter 2 to the case when the constrains are given by piece-wise linear functions. We derive the optimality conditions and the solution algorithm (Algorithm 2) for this problem. Finally, we show that under some nondegeneracy assumptions Algorithm 2 terminates in a finite number of steps (Theorem 3.3).

In Chapter 4 we show how spline approximation can be used to transform the problem (1.3) into a form amenable to the interior point methods. We also establish the continuity of the spline approximation which means that (1.3) can be solved with any desired accuracy by selecting sufficiently accurate spline approximations.

Computational results and concluding remarks are given in Chapters 5 and 6 respectively.

Chapter 2

The Active Set Algorithm

2.1 Problem Formulation

We consider the problem of minimization of the function $f(x)$ subject to linear inequality constraints.

$$(P) \qquad \begin{array}{ll} \min & f(x) \\ \text{s.t.} & Ax \leq b, \end{array} \tag{2.1}$$

where A is an $m \times n$-matrix and $b \in \mathbb{R}^m$. Let us denote the rows of the matrix A by $(a^k)^T,\ k = 1, \ldots, m$. The objective function $f(x)$ is defined as follows:

$$f(x) = F(x) + \sum_{i=1}^{n} f_i(x_i). \tag{2.2}$$

We require the following assumption to be satisfied throughout the thesis.

Assumption 2.1

1. *$F(x)$ is a convex continuously differentiable function;*

2. *f_i, $i = 1, \ldots, n$ are convex functions of the following form:*

$$f_i(x_i) = \begin{cases} f_{i0}(x_i), & \text{if} \quad x_i \leq d_{i1}, \\ f_{il}(x_i), & \text{if} \quad d_{il} \leq x_i \leq d_{il+1}, \quad l = 1, \ldots, M_i, \\ f_{iM_i}(x_i), & \text{if} \quad x_i \geq d_{iM_i}; \end{cases} \tag{2.3}$$

3. *f_{il} are convex continuously differentiable on (d_{il}, d_{il+1}) and have one-sided derivtives at the ends of the intervls; f_{i0} are convex continuously differentiable on $(-\infty, d_{i1})$ and have left derivtives at d_{i1} ; f_{iM_i} are convex continuously differentiable on $(d_{iM_i}, +\infty)$ and have right derivtives at d_{iM_i} for each $i = 1, \ldots, n$, $l = 0, \ldots, M_i$.*

Let the feasible region of the problem (2.1) be denoted by S; and, for each $x \in \mathbb{R}^n$, let the set of *active breakpoints* be denoted by

$$E(x) = \{i : x_i = d_{il} \text{ for some } l \in \{1, .., M_i\}\}, \quad N(x) = \{1, \ldots, n\} \backslash E(x). \tag{2.4}$$

2.2 Duality and Optimality

The Lagrangian dual of (P) is

$$\max_{u \geq 0} \min_{x} L(x, u) := f(x) + u^T(Ax - b).$$

The inner-minimization is an unconstrained convex minimization problem. Therefore, we can write down the Wolfe dual program

$$(D) \qquad \begin{array}{ll} \max & L(x, u) \\ \text{s.t.} & 0 \in \partial_x L(x, u), \\ & u \geq 0, \end{array} \tag{2.5}$$

where $\partial_x L(x,u)$ denotes the subgradient of L, i.e.

$$\partial_x L(x,u) = \{\phi \in \mathbb{R}^n : \phi^T(y-x) \leq L(y,u) - L(x,u), \quad \forall y \in \mathbb{R}^n\}.$$

We can now state the well-known optimality conditions.

Theorem 2.1 *A point $x \in \mathbb{R}^n$ minimizes f over S if and only if the following system holds*

$$\begin{array}{rl} u \geq 0, \quad 0 \in \partial_x L(x,u) & \textit{dual feasibility} \\ Ax \leq b & \textit{primal feasibility} \\ u^T(Ax-b) = 0 & \textit{complementary slackness.} \end{array}$$

■

To further simplify the optimality conditions, we use the following property of subgradients:

Proposition 2.1 *Let $\theta = \sum_{i=1}^m \theta_i$, where $\theta_i : \mathbb{R}^n \to \mathbb{R}$ are convex functions, $i = 1, \ldots, m$. Then $\partial\theta(x)$ is equal to the Minkowski sum*

$$\sum_{i=1}^m \partial\theta_i(x).$$

Proof. See [4]. ■

Recall that

$$L(x,u) := F(x) + \sum_{i=1}^n f_i(x_i) + u^T(Ax-b).$$

Since the first and the last terms of this sum are differentiable, the subgradient is equal to the gradient for these functions.

Therefore, for the Lagrangian $L(x,u)$

$$\partial L(x,u) = \nabla F(x) + \partial(\sum_{i=1}^{n} f_i(x_i)) + A^T u.$$

It follows from the definition of $f_i(x_i)$ that

$$\partial f_i(x_i) = \begin{cases} \frac{df_{i0}}{dx_i}(x_i), & \text{if} \quad x_i < d_{i1}, \\ \frac{df_{il}}{dx_i}(x_i), & \text{if} \quad d_{il} < x_i < d_{il+1}, \ \ l = 1, ..., M_i, \\ [\frac{df_{il-1}}{dx_i(x_i)}, \frac{df_{il}}{dx_i}(x_i)] & \text{if} \quad x_i = d_{il}, \ \ l = 1, ..., M_i, \\ \frac{df_{iM_i}}{dx_i}(x_i), & \text{if} \quad x_i > d_{iM_I}. \end{cases}$$

We can think of f_i as a function from $\mathbb{R}^n$ to $\mathbb{R}$, defined as $f_i(x) = f_i(x_i)$. Then $\partial f_i(x) = \partial f_i(x_i)e_i$ and $\sum_{i=1}^n f_i(x_i) = \sum_{i=1}^n f_i(x)$. By Proposition 2.1, $\partial(\sum_{i=1}^n f_i(x_i))$ is equal to the Minkowski sum $\sum_{i=1}^n \partial f_i(x_i)e_i$. From the definition of the Minkowski sum, the latter sum is equal to a direct product of $\partial f_i(x_i)$. Therefore,

$$0 \in \partial L(x,u)$$

if and only if, for every $i = 1, \ldots, n$,

$$0 \in (\nabla F(x))_i + \partial f_i(x_i) + (A^T u)_i.$$

This allows us to reformulate the optimality conditions for (2.1).

Corollary 2.1 *Let the function f be given by (2.2). A point $x \in \mathbb{R}^n$ minimizes f*

over S, if and only if there exists $u \in \mathbb{R}^m$, $v \in \mathbb{R}^{E(x)}$ such that

$$\left.\begin{array}{ll}
Ax \le b, & \\
(\nabla F(x))_i + \frac{df_{il}}{dx_i}(x_i) + (A^T u)_i = 0, & \text{for all } i \in N(x), \\
 & \text{with } x_i \in (d_{il}, d_{il+1}), \\
(\nabla F(x))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + (A^T u)_i + v_i = 0, & \text{for all } i \in E(x), \\
 & \text{with } x_i = d_{il}, \\
0 \le v_i \le \frac{df_{il}}{dx_i}(d_{il}) - \frac{df_{il-1}}{dx_i}(d_{il}), & \text{for all } i \in E(x), \\
 & \text{with } x_i = d_{il}, \\
u \ge 0, & \\
u^T(Ax - b) = 0. &
\end{array}\right\} \qquad (2.6)$$

2.3 Algorithm 1

In this section, we formulate an algorithm for the solution of the problem (2.1). As always, we assume that the function $f(x)$ is given by (2.2) and Assumption 2.1 is satisfied.

Let x be feasible for the problem (2.1). Then the function $D(x) = (D_1(x_1), \ldots, D_n(x_n))$ given by

$$D_i(x_i) = l \text{ if } x_i \in [d_{il}, d_{il-1}),$$

will be called the **interval indicator function**. In other words, if we subdivide the domain of the function $f(x)$ into n-dimensional blocks by the hyperplanes $x_i = d_{il}$, $l = 1, \ldots, M_i$, $i = 1, \ldots, n$ then knowing $D(x)$ is equivalent to knowing in which of these n-dimensional blocks x is located.

The main idea behind the algorithm comes from the optimality conditions (2.6). Suppose x^* is the optimal solution of (2.1). If we guessed the correct active set $I(x^*)$,

set of breakpoints $E(x^*)$ and the interval indicator function $D(x^*)$, the problem would be reduced to an equality constrained problem.

$$\begin{aligned} \min \quad & f(x) \\ \text{s. t.} \quad & a^k x = b^k, \ k \in I(x^*), \\ & x_i = d_{il}, \ i \in E(x^*), \text{ with } l = D(x_i^*)_i. \end{aligned} \tag{2.7}$$

Similarly to the active set method, we are going to search through a sequence of sets to identify the likely set of active constraint indices $I(x^*)$, set $E(x^*)$ and the interval indicator function $D(x^*)$ at the optimum.

We start with a point x^0 which is feasible for (2.1). We then choose a subset $I^0 \subseteq I(x^0)$ of constraints active at x^0, and a subset $E^0 \subseteq E(x^0)$ of coordinates of x^0 that are at break points, such that the vectors $\{a^k, \ k \in I^0, e_i, \ i \in E^0\}$ are linearly independent. We also set $D^0 = D(x^0)$. We will refer to I^0, E^0, and D^0 as a working active set, a working break point set, and a working interval indicator function respectively.

For $j \geq 1$, the initial point in iteration j is x^j, the feasible point obtained at the end of iteration $j-1$. We'll denote the working active set in the iteration j by I^j, the working set of breakpoints by E^j, and the working interval indicator vector by D^j. We will show how these I^j, E^j and D^j are constructed when we give a detailed description of the algorithm. The next feasible point will be selected within the n-dimensional block defined by D^j. Note that, the following conditions will be satisfied at each iteration

$$\begin{aligned} & I^j \subseteq I(x^j), \ E^j \subseteq E(x^j), \\ & \{a^k, \ k \in I^j, e_i, \ i \in E^j\} \quad \text{are linearly independent,} \\ & D_i(x^j) - 1 \leq D_i^j \leq D_i(x^j). \end{aligned}$$

The objective function in iteration j is defined by

$$f^j(x) = F(x) + \sum_{i=1}^{n} f_{i,D_i^j}(x_i).$$

As discussed in Appendix B, the objective function of the original problem coincides with $f^j(x)$ on the n-dimensional block, corresponding to D^j.

In iteration j, we are solving the equality constrained minimization problem

$$\begin{aligned} \min \quad & f^j(x) \\ \text{s. t.} \quad & a^k x = b^k, \ k \in I^j, \\ & x_i = d_{il}, \ i \in E^j, \ \text{with } l = D_i^j. \end{aligned} \tag{2.8}$$

Let s^j be the search direction obtained at x^j, i.e. $y^j = x^j + s^j$ is feasible for (2.8) and $f^j(y^j) < f^j(x^j)$. Note that (2.8) is a differentiable convex linearly constrained problem. In many cases it can be solved in a finite number of steps. In such cases we will select y^j, which is optimal for (2.8). We have two possibilities to consider:

(a) A feasible descent direction $s^j \neq 0$ is found for (2.8) using, for example, projected gradient or projected Hessian method, see section 1.2.

(b) Vector x^j satisfies the first order optimality conditions for (2.8) and therefore no feasible descent direction can be found for this problem.

Let us consider case **(a)** first. Here we can perform a line search to minimize $f^j(x^j + \sigma s^j)$ over $\sigma \geq 0$. Let $\hat{\sigma}$ be the optimal step size for this problem. Ideally, we would like to take $x^{j+1} = x^j + \hat{\sigma} s^j$ as the next approximation. However the point $x^j + \hat{\sigma} s^j$ could be infeasible for the original problem or it could be outside of the n-dimensional block defined by D^j. In this case we want to find the value of σ such that $x^j + \sigma s^j$ is feasible, stays within the block and gives the best possible objective function value under these conditions.

To do this we compute

$$\sigma_0 = \frac{b^{i_0} - (a^{i_0})^T x^j}{(a^{i_0})^T s^j} = \min\left\{ \frac{b^k - (a^k)^T x^j}{(a^k)^T s^j} \ : \ k \notin I^j \text{ and } (a^k)^T s^j > 0 \right\}, \tag{2.9}$$

$$\sigma_1 = \frac{d_{i_1 l} - x^j_{i_1}}{s^j_{i_1}} = \min\left\{ \frac{d_{il} - x^j_i}{s^j_i} \ : \ i \notin E^j \text{ and } s^j_i < 0, \ l = D^j_i \right\}, \tag{2.10}$$

$$\sigma_2 = \frac{d_{i_2 l+1} - x^j_{i_2}}{s^j_{i_2}} = \min\left\{ \frac{d_{il+1} - x^j_i}{s^j_i} \ : \ i \notin E^j \text{ and } s^j_i > 0, \ l = D^j_i \right\}. \tag{2.11}$$

In the above, i_0, i_1, i_2 are the indices for which the minima occur in the calculation of $\sigma_0, \sigma_1, \sigma_2$, respectively. Let

$$\bar{\sigma} = \min\{\sigma_0, \sigma_1, \sigma_2\}.$$

If no σ_0, σ_1 or σ_2 were calculated, we set $\bar{\sigma} = +\infty$. We determine the j-th step size by

$$\sigma^j = \min\{\bar{\sigma}, \hat{\sigma}\}.$$

If $\sigma^j = +\infty$ we conclude that the the problem is unbounded from below and stop. Otherwise we set

$$x^{j+1} = x^j + \sigma^j s^j.$$

If $\sigma^j < \bar{\sigma}$, we leave the working active set I^j, the working set of breakpoints E^j and the working interval indicator vector D^j unchanged, and proceed to the next step.

Now consider the case when $\sigma^j = \bar{\sigma}$. Clearly, one of the following conditions must hold true: $\bar{\sigma} = \sigma_0$, $\bar{\sigma} = \sigma_1$, or $\bar{\sigma} = \sigma_2$.

(i) If $\bar{\sigma} = \sigma_0$ then all the indices i which tie in the minimum of the definition of σ_0 join the active set $I(x^{j+1})$; we choose one of them, say i_0 and include it in the working active set: $I^{j+1} = I^j \cup \{i_0\}$.

(ii) If $\bar{\sigma} = \sigma_1$ then all the indices i which tie in the minimum of the definition of σ_1 join the set of breakpoints $E(x^{j+1})$; we choose one of them, say i_1 and include it in the working set of breakpoints: $E^{j+1} = E^j \cup \{i_1\}$. Note that $x_{i_1}^{j+1} = d_{i_1 l}$, where $l = D_{i_1}^{j+1} = D_{i_1}(x^{j+1})$.

(iii) If $\bar{\sigma} = \sigma_2$ then all the indices i which tie in the minimum of the definition of σ_2 join the set of breakpoints $E(x^{j+1})$; we choose one of them, say i_2 and include it in the working set of breakpoints: $E^{j+1} = E^j \cup \{i_2\}$. We also update the corresponding indicator function $D_{i_2}^{j+1} = D_{i_2}^j + 1$. Note that $x_{i_2}^{j+1} = d_{i_1 l+1}$, where $l + 1 = D_{i_2}^{j+1} = D_{i_2}(x^{j+1})$.

Finally, if there is a tie in the minimum of the definition of $\bar{\sigma}$, we only perform one of the applicable steps (i)-(iii).

Now, let us consider case **(b)**. Recall that in this case vector x^j satisfies the first order optimality conditions for (2.8) and therefore no feasible descent direction can be found for this problem.

For $i \in I^j$ let u_i^j be the Lagrange multiplier corresponding to the constraint $a_i^T x = b_i$ of (2.8). If $i \notin I^j$ we set $u_i^j = 0$. Similarly, for $i \in E^j$ let λ_i^j be the Lagrange multiplier corresponding to the constraint $x_i = d_{il}$. In order to check if the point x^j is optimal for the original problem, we need to calculate the following values for each $i \in E^j$.

$$\begin{aligned} v_i^j &= -(\nabla F(x^j))_i - \tfrac{df_{il-1}}{dx_i}(x_i^j) - (A^T u^j)_i, \\ w_i^j &= (\nabla F(x^j))_i + \tfrac{df_{il}}{dx_i}(x_i^j) + (A^T u^j)_i = -\lambda_i^j, \end{aligned} \tag{2.12}$$

where $l = D_i^j$. If $u_i^j \geq 0$ for all $i \in I^j$, $v_i^j \geq 0$ and $w_i^j \geq 0$ for all $i \in E^j$, we conclude that the optimality conditions given by Corollary 2.1 are also satisfied by x^j. The algorithm terminates with the optimal solution x^j.

Otherwise we calculate

$$u_{k_0} = \min_{i \in I^j}\{u_i^j\},$$
$$v_{k_1} = \min_{i \in E^j}\{v_i^j\},$$
$$w_{k_2} = \min_{i \in E^j}\{w_i^j\}$$

and let

$$\bar{u} = \min\{u_{k_0}, v_{k_1}, w_{k_2}\}.$$

Naturally, $\bar{u}$ must be equal to at least one of u_{k_0}, v_{k_1}, or w_{k_2}.

(i) If $\bar{u} = u_{k_0}$ then delete k_0 from the working active set: $I^{j+1} = I^j - \{k_0\}$;

(ii) If $\bar{u} = v_{k_1}$ then delete k_1 from the working set of breakpoints: $E^{j+1} = E^j - \{k_1\}$ and change $D_{k_1}^{j+1} = D_{k_1}^{j+1} - 1$.

(iii) If $\bar{u} = w_{k_2}$ then delete k_2 from the working set of breakpoints: $E^{j+1} = E^j - \{k_2\}$.

Again, if there is a tie in the minimum of the definition of $\bar{u}$, we only perform one of the applicable steps (i)-(iii). Note that updates associated with changing the working active set are computationally more expensive than the ones associated with the working set of breakpoints, so we do steps (ii) or (iii) in case of a tie. This ends the description of case **(b)**.

Remark 2.1 *As a special case of Algorithm 1, one can derive a similar algorithm for the unconstrained minimization of (2.2).*

Remark 2.2 *In order to simplify the notation we set $d_{i0} = -\infty$, $d_{iM_i+1} = +\infty$ for all $i = 1, ..., n$. This allows us to reduce the number of special cases that have to be considered.*

Let us now summarize the algorithm for solving the linearly constrained problem (2.1).

Algorithm 1

Model Problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s. t.} \quad & a^k x \leq b^k \ k = 1, \ldots, m, \end{aligned}$$

where $f(x)$ is defined by (2.2) and Assumption 2.1 is satisfied.

Initialization Begin with any $x^0 \in S$.

For each $i = 1, \ldots, n$ define the working interval indicator $D_i^0 = l$ if $x_i^0 \in [d_{il}, d_{il+1})$.

Chose an initial working set of breakpoints E^0 such that $E^0 \subseteq E(x^0)$ and the vectors $\{a^k, \ k \in I^0, e_i, \ i \in E^0\}$ are linearly independent.

Set $j = 0$.

Step 1: Computation of Search Direction

Define the objective function f^j for the j-th subproblem

$$f^j(x) = F(x) + \sum_{i=1}^n f_{i,D_i^j}(x_i).$$

Step1.1

Consider the equality constrained minimization problem

$$\begin{aligned} \min \quad & f^j(x) \\ \text{s. t.} \quad & (a^k)^T x = b^k \ k \in I^j, \\ & x_i = d_{il}, \ i \in E^j, \ \text{with } l = D_i^j. \end{aligned} \tag{2.13}$$

If x^j is a stationary point for (2.13), for all $k \in I^j$ take u_k^j to be the Lagrange multiplier for the constraint $(a^k)^T x = b^k$, for all $i \in E^j$ take λ_i^j to be the Lagrange multiplier for the constraint $x_i = d_{il}$ of (2.13), and proceed to Step 3.2.

Otherwise, find y^j such that it is feasible for (2.13) and $f^j(y^j) < f^j(x^j)$. Set $s^j = y^j - x^j$ and go to Step 2.

Step 2: Computation of the Step Size

Compute the smallest indices i_1, i_2 and i_3 such that

$$\sigma_0 = \frac{b^{i_0} - (a^{i_0})^T x^j}{(a^{i_0})^T s^j} = \min\left\{ \frac{b^k - (a^k)^T x^j}{(a^k)^T s^j} \ : \ k \notin I^j \text{ and } (a^k)^T s^j > 0 \right\}, \quad (2.14)$$

$$\sigma_1 = \frac{d_{i_1 l} - x_{i_1}^j}{s_{i_1}^j} = \min\left\{ \frac{d_{il} - x_i^j}{s_i^j} \ : \ i \notin E^j \text{ and } s_i^j < 0, \ l = D_i^j \right\}, \quad (2.15)$$

$$\sigma_2 = \frac{d_{i_2 l+1} - x_{i_2}^j}{s_{i_2}^j} = \min\left\{ \frac{d_{il+1} - x_i^j}{s_i^j} \ : \ i \notin E^j \text{ and } s_i^j > 0, \ l = D_i^j \right\}. \quad (2.16)$$

If none of σ_0, σ_1 nor σ_2 were calculated, set $\bar{\sigma} = +\infty$, otherwise set

$$\bar{\sigma} = \min\{\sigma_0, \sigma_1, \sigma_2\}.$$

Perform a line search to minimize $f^j(x^j + \sigma s^j)$ over $0 \leq \sigma \leq \bar{\sigma}$. Let σ^j be the optimal step length for this problem.

If $\sigma^j = +\infty$, stop, the problem is unbounded from below. Otherwise, go to Step 3.1.

Step 3: Update

Step 3.1 Set

$$x^{j+1} = x^j + \sigma^j s^j.$$

If $\sigma^j < \bar{\sigma}$, set $I^{j+1} = I^j$, $E^{j+1} = E^j$, $D^{j+1} = D^j$, $j = j + 1$ and go to Step 1.

Otherwise

(i) If $\bar{\sigma} = \sigma_0$ set $I^{j+1} = I^j \cup \{i_0\}$, $E^{j+1} = E^j$, $D^{j+1} = D^j$, $j = j + 1$; go to Step 1.

(ii) If $\bar{\sigma} = \sigma_1$ set $I^{j+1} = I^j$, $E^{j+1} = E^j \cup \{i_1\}$, $D^{j+1} = D^j$, $j = j + 1$; go to Step 1.

(iii) If $\bar{\sigma} = \sigma_2$ set $I^{j+1} = I^j$, $E^{j+1} = E^j \cup \{i_2\}$ and $D^{j+1}_{i_2} = D^j_{i_2} + 1$, $D^{j+1}_i = D^j_i$ for all $i \neq i_2$, $j = j + 1$; go to Step 1.

Step 3.2 Set

$$x^{j+1} = x^j.$$

For each $i \in E^j$ calculate

$$v^j_i = -(\nabla F(x^j))_i - \frac{df_{il-1}}{dx_i}(x^j_i) - \sum_{k \in I^j} a^k_i u^j_k, \tag{2.17}$$

$$w^j_i = (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x^j_i) + \sum_{k \in I^j} a^k_i u^j_k, \tag{2.18}$$

where $l = D^j_i$.

If $u^j_k \geq 0$ for all $k \in I^j$, $v^j_i \geq 0$ and $w^j_i \geq 0$ for all $i \in E^j$, stop with optimal solution x^j.

Otherwise, calculate

$$u_{k_0} = \min_{k \in I^j}\{u^j_k\},$$
$$v_{k_1} = \min_{i \in E^j}\{v^j_i\},$$
$$w_{k_2} = \min_{i \in E^j}\{w^j_i\}.$$

Set

$$\bar{u} = \min\{u_{k_0}, v_{k_1}, w_{k_2}\}.$$

(i) If $\bar{u} = u_{k_0}$ set $I^{j+1} = I^j - \{k_0\}$, $E^j = E^j$, $D^j = D^j$, $j = j + 1$; go to Step 1.

(ii) If $\bar{u} = v_{k_1}$ set $I^{j+1} = I^j$, $E^{j+1} = E^j - \{k_1\}$ and $D^{j+1}_{k_1} = D^j_{k_1} - 1$, $D^{j+1}_i = D^j_i$ for all $i \neq k_1$, $j = j + 1$; go to Step 1.

(iii) If $\bar{u} = w_{k_2}$ set $I^{j+1} = I^j$, $E^{j+1} = E^j - \{k_2\}$, $D^{j+1} = D^j$, $j = j + 1$; go to Step 1.

Example 2.1 *Consider (2.1) with*

$$F(x) = x_1^2 + x_2^2 - 2x_1 - 6x_2,$$

$$f_1(x) = \begin{cases} -0.2x & -0.2, \text{ if } x \leq -2, \\ -0.1x & \quad , \text{ if } x \in [-2, 0], \\ 0.1x & \quad , \text{ if } x \in [0, 2], \\ 0.2x & -0.2, \text{ if } x \geq 2, \end{cases}$$

and

$$f_2(x) = \begin{cases} -0.1x & -0.2, \text{ if } x \leq -2, \\ 0 & \quad , \text{ if } x \in [-2, 0], \\ 0.1x & \quad , \text{ if } x \in [0, 2], \\ 0.2x & -0.2, \text{ if } x \geq 2. \end{cases}$$

We solve the problem

$$\min\{f(x) \mid x_1 + x_2 \leq 3\},$$

where $f(x) = F(x) + f_1(x_1) + f_2(x_2)$.

Initialization.

Choose $x_0 = (0,0)^T$ to be the starting point. Set $D_1^0 = 3,\ D_2^0 = 3$, $I^0 = \emptyset$, and $E^0 = \{1,2\}$.

Iteration 0.

Step 1. Set $f^0 = F(x) + 0.1x_1 + 0.1x_2$.

Step 1.1. Solve the subproblem

$$\min\{f^0(x) \,|x_1 = 0, x_2 = 0\}.$$

The optimal solution of the subproblem is $(0,0)^T$, $s^0 = (0,0)^T$.

Step 3. Set $x^1 = x^0 = (0,0)^T$. The corresponding value of f is $f(x^1) = 0$.

Step 3.2. Calculate $v = (2.1, 6)^T$ and $w = (-1.9, -5.9)^T$. The smallest of these numbers is $w_2 = -5.9$. Therefore we set $D_1^0 = 3,\ D_2^0 = 3$, $I^0 = \emptyset$, and $E^0 = \{1\}$.

Iteration 1.

Step 1. Set $f^1 = F(x) + 0.1x_1 + 0.1x_2$.

Step 1.1. Solve the subproblem

$$\min\{f^1(x) \,|x_1 = 0\}.$$

The optimal solution of this subproblem is $(0, 2.95)^T$, $s^1 = (0, 2.95)^T$.

Step 2. Compute $\hat{\sigma} = 2/2.95$. $\bar{\sigma} = 1$. Set $\sigma_1 = \hat{\sigma} = 2/2.95$.

Step 3. Set $x^2 = (0,0)^T + 2/2.95(0, 2.95)^T = (0,2)^T$. The corresponding value of f is $f(x^2) = -7.8$.

Step 3.1. Set $D_1^0 = 3,\ D_2^0 = 4$, $I^0 = \emptyset$, and $E^0 = \{1,2\}$.

Iteration 2.

Step 1. Set $f^2 = F(x) + 0.1x_1 + 0.2x_2 - 0.2$.

Step 1.1. Solve the subproblem

$$\min\{f^2(x) \mid x_1 = 0,\ x_2 = 2\}.$$

The optimal solution of this subproblem is $(0,2)^T$, $s^2 = (0,0)^T$.

Step 3. Set $x^3 = x^2 = (0,2)^T$. The corresponding value of f is $f(x^3) = -7.8$.

Step 3.2. Calculate $v = (2.1, 1.9)^T$ and $w = (-1.9, -1.8)^T$. The smallest of these numbers is $w_1 = -1.9$. Therefore we set $D_1^0 = 3,\ D_2^0 = 4$, $I^0 = \emptyset$, and $E^0 = \{2\}$.

Iteration 3.

Step 1. Set $f^3 = F(x) + 0.1x_1 + 0.2x_2 - 0.2$.

Step 1.1. Solve the subproblem

$$\min\{f^3(x) \mid 0 \le x_1 \le 2,\ x_2 = 2\}.$$

The optimal solution of this subproblem is $(0.95,2)^T$, $s^3 = (1,0)^T$.

Step 2. Compute $\hat{\sigma} = 1$. $\bar{\sigma} = 0.95$. Set $\sigma_3 = \bar{\sigma} = 0.95$.

Step 3. Set $x^4 = (0,2)^T + 0.95(1,0)^T = (0.95,2)^T$. The corresponding value of f is $f(x^4) = -8.7025$.

Set $D_1^0 = 3,\ D_2^0 = 4$, $I^0 = \emptyset$, and $E^0 = \{2\}$.

Iteration 4.

Step 1. Set $f^4 = F(x) + 0.1x_1 + 0.2x_2 - 0.2$.

Step 1.1. Solve the subproblem

$$\min\{f^4(x) \mid 0 \le x_1 \le 2,\ x_2 = 2\}.$$

The optimal solution of this subproblem is $(0.95,2)^T$, $s^4 = (0,0)^T$.

Step 3. Set $x^5 = x^4 = (0.95, 2)^T$. The corresponding value of f is $f(x^5) = -8.7025$.

Step 3.2. Calculate $v_2 = 0$ and $w_2 = -1.8$. The smallest of these numbers is w_2. Therefore we set $D_1^0 = 3,\ D_2^0 = 4$, $I^0 = \emptyset$, and $E^0 = \emptyset$.

Iteration 5.

Step 1. Set $f^5 = F(x) + 0.1x_1 + 0.2x_2 - 0.2$.

Step 1.1. Solve the subproblem

$$\min\{f^5(x)\}.$$

The optimal solution of this subproblem is $(0.95, 2.9)^T$, $s^5 = (0, 1)^T$.

Step 2. Compute $\hat{\sigma} = 0.9$. $\bar{\sigma} = 0.05$. Set $\sigma_5 = \bar{\sigma} = 0.05$.

Step 3. Set $x^6 = (0.95, 2)^T + 0.05(0.1)^T = (0.95, 2.05)^T$. The corresponding value of f is $f(x^6) = -8.7900$.

Set $D_1^0 = 3,\ D_2^0 = 4$, $I^0 = \{1\}$, and $E^0 = \emptyset$.

Iteration 6.

Step 1. Set $f^6 = F(x) + 0.1x_1 + 0.2x_2 - 0.2$.

Step 1.1. Solve the subproblem

$$\min\{f^6(x) \mid x_1 + x_2 = 3\}.$$

The optimal solution of this subproblem is $(0.525, 2.475)^T$, $s^6 = (-0.425, 0.425)^T$.

Step 2. Compute $\sigma_6 = \hat{\sigma} = 1$.

Step 3. Set $x^7 = (0.95, 2.05)^T + (-0.425, 0.425)^T = (0.525, 2.475)^T$. The corresponding value of f is $f(x^7) = -9.15125$.

Set $D_1^0 = 3,\ D_2^0 = 4$, $I^0 = \{1\}$, and $E^0 = \emptyset$.

Iteration 7.

Step 1. Set $f^7 = F(x) + 0.1x_1 + 0.2x_2 - 0.2$.

Step1.1. Solve the subproblem

$$\min\{f^7(x) \mid x_1 + x_2 = 3\}.$$

The optimal solution of this subproblem is $(0.525, 2.475)^T$, $s^7 = (0,0)^T$.

Step 3.2. No vs or ws need to be calculated. We have $u^7 = 0.85 > 0$ and therefore x^7 is the optimal solution.

Figure 2.3 shows the feasible region of the example problem and the points, generated by Algorithm 1 $x^0, \ldots, x^7$.

Note that the sequence of objective function values $f(x^0), \ldots, f(x^7)$ is decreasing.

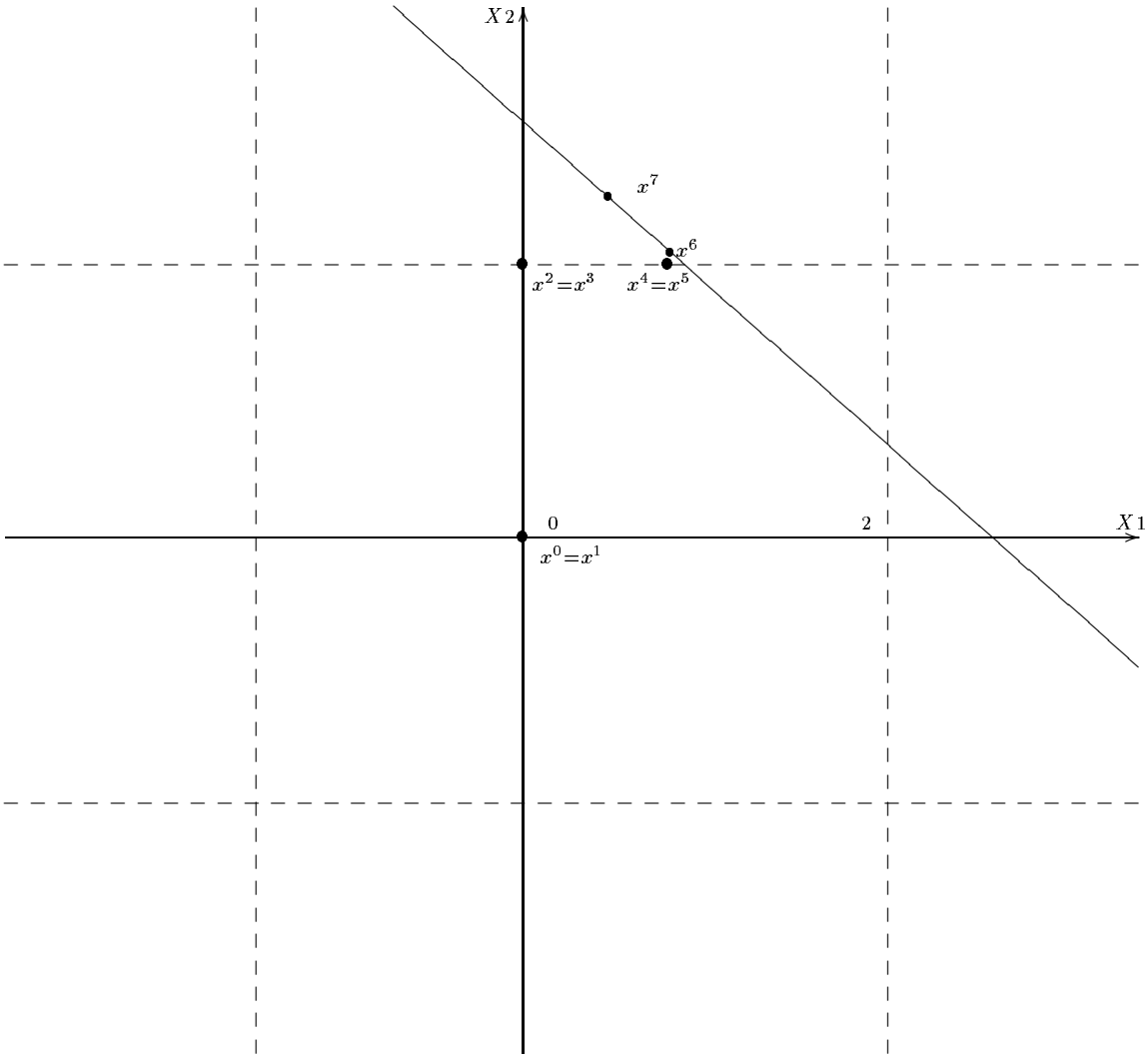

Figure 2.1: Progress of Algorithm 1 in Solving Example 2.1.

Chapter 3

Piece-wise Linear Inequality Constraints

3.1 Optimality Conditions

Many of the constraints that arise in practice are not linear. For example, to impose an upper bound on the amount bought we can use one constraint $g(x) \leq K$, where $g(x) = \sum_{i=1}^{n} g_i(x_i)$ and

$$g_i(x_i) = \begin{cases} 0, & \text{if} \quad x_i < \hat{x}_i, \\ x_i - \hat{x}_i, & \text{if} \quad x_i \geq \hat{x}_i. \end{cases}$$

In portfolio optimization this is called a turnover constraint.

In this section, we generalize results of the previous section to the problem of minimization of (2.2) subject to piecewise linear convex inequality constraints. We consider the problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g^k(x) \leq h^k, \ \ k = 1, \ldots, m, \end{array} \tag{3.1}$$

where function $f(x)$ is given by (2.2), and Assumption 2.1 is satisfied.

We further require that each $g^k(x)$ is a piecewise linear convex function of the form $g^k(x) = \sum_{i=1}^n g_i^k(x_i)$,

$$g_i^k(x_i) = \begin{cases} a_{i0}^k x_i + b_{i0}^k, & \text{if} \quad x_i \le d_{i1}, \\ a_{il}^k x_i + b_{il}^k, & \text{if} \quad x_i \in [d_{il}, d_{il+1}], \ \ l = 1, \ldots, M_i, \\ a_{iM_i}^k x_i + b_{iM_i}^k, & \text{if} \quad x_i \ge d_{iM_i}. \end{cases} \tag{3.2}$$

Without loss of generality we can assume that M_i, $i = 1, \ldots, n$ and d_{il}, $i = 1, \ldots, n$, $l = 1, \ldots, M_i$ are the same as in the definition of the functions $f_i(x_i)$.

Since $g^k(x)$ is a convex function, $g_i^k(x_i)$ is also convex. On the other hand, it follows from Proposition B.1 that for every $k = 1, \ldots, m$, $i = 1, \ldots, n$

$$a_{il-1}^k d_{il} + b_{il-1}^k = a_{il}^k d_{il} + b_{il}^k \text{ for each } \ \ l = 1, \ldots, M_i, \tag{3.3}$$

and

$$a_{il-1}^k \le a_{il}^k \text{ for each } \ \ l = 1, \ldots, M_i. \tag{3.4}$$

Lemma 3.1 *Let $g_i^k(x_i)$ be a piecewise linear convex function defined by (3.2). Then for any $x_i \in \mathbb{R}$*

$$g_i^k(x_i) = \max_{l=0,\ldots,M_i} a_{il}^k x_i + b_{il}^k.$$

Moreover,

(i) for any $l = 1, \ldots, M_i$ and for any x_i satisfying $d_{il} < x_i < d_{il+1}$,

$$g_i^k(x_i) = a_{ij}^k x_i + b_{ij}^k$$

if and only if $a_{ij}^k = a_{il}^k$;

(ii) for any $l = 1, \ldots, M_i$,

$$g_i^k(d_{il}) = a_{ij}^k d_{il} + b_{ij}^k$$

if an only if $a_{ij}^k = a_{il}^k$ *or* $a_{ij}^k = a_{il-1}^k$.

We prove this lemma in Appendix A.

As is stated below, it is possible to reformulate problem (3.1) as a linearly constrained problem. However this would significantly increase the number of constraints and variables.

Lemma 3.2 *Consider the problem*

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & \sum_{i=1}^n y_i^k \leq h^k \;\; k = 1, \ldots, m \\ & y_i^k \geq a_{il}^k x_i + b_{il}^k \quad k = 1, \ldots, m, \;\; i = 1, \ldots, n, \;\; l = 0, \ldots, M_i. \end{array} \tag{3.5}$$

Then $x^* \in \mathbb{R}^n$ *is optimal for (3.1) if and only if* (x^*, y^*) *is optimal for (3.5) for some* $y^* \in \mathbb{R}^{nm}$.

Proof of this lemma is given in Appendix A.

The above representation can be used to obtain the optimality conditions for (3.1).

Theorem 3.1 *A vector* $x^* \in \mathbb{R}^n$ *is optimal for (3.1) if and only if there exists*

$u^* \in \mathbb{R}^m$ *such that*

$$\left.\begin{array}{ll}
g^k(x^*) \le h^k, & k = 1, \ldots, m, \\
(\nabla F(x^*))_i + \frac{df_{il}}{dx_i}(x_i^*) + \sum_{k=1}^m a_{il}^k u_k^* = 0, & \text{for all } i \in N(x^*), \\
 & \text{with } x_i^* \in (d_{il}, d_{il+1}), \\
(\nabla F(x^*))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il-1}^k u_k^* \le 0, & \text{for all } i \in E(x^*), \\
 & \text{with } x_i^* = d_{il}, \\
(\nabla F(x^*))_i + \frac{df_{il}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il}^k u_k^* \ge 0, & \text{for all } i \in E(x^*), \\
 & \text{with } x_i^* = d_{il}, \\
u^* \ge 0, & \\
u_k^*(g^k(x^*) - h^k) = 0, & k = 1, \ldots, m.
\end{array}\right\} \tag{3.6}$$

Proof. Let us denote the feasible region of the problem (3.1) by S. Suppose that (3.6) is satisfied for some $u^* \in \mathbb{R}^m$. The Lagrangian function

$$L(x, u^*) = f(x) + \sum_{k=1}^m g^k(x) u_k^*$$

is a convex function of x, since it is a linear combination of convex functions with nonnegative coefficients. The optimality conditions (3.6) imply that x^* minimizes $L(x, u^*)$ over $\mathbb{R}^n$. Therefore x^* minimizes $f(x)$ over S.

To prove the converse, suppose that x^* is an optimal solution for (3.1). Define y^* by

$$(y^*)_i^k = g_i^k(x^*) = \max\{a_{ij}^k x_i^* + b_{ij}^k, \;\; j = 0, \ldots, M_i\}.$$

The vector (x^*, y^*) is clearly feasible for (3.5). So, by Lemma 3.2, (x^*, y^*) is optimal for problem (3.5).

Note that the problem (3.5) is a linearly constrained problem. Therefore by

Corollary 2.1, there exist vectors $\bar{u}$ and λ such that the following holds true.

$$\left.\begin{array}{lll}
\sum_{i=1}^{n}(y^*)_i^k \le h^k, \;\; k=1,\ldots,m, & & \\
(y^*)_i^k \ge a_{ij}^k x_i^* + b_{ij}^k \;\; i=1,\ldots,n, \;\; j=0,\ldots,M_i, & k=1,\ldots,m, & \\
(\nabla F(x^*))_i + \frac{df_{il}}{dx_i}(x_i^*) + \sum_{k=1}^{m}\sum_{j=0}^{M_i} a_{ij}^k\lambda_{ij}^k = 0, & \text{for all } i \in N(x^*), & \\
 & \text{with } x_i^* \in (d_{il}, d_{il+1}), & \\
(\nabla F(x^*))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + \sum_{k=1}^{m}\sum_{j=0}^{M_i} a_{ij}^k\lambda_{ij}^k \le 0, & \text{for all } i \in E(x^*), & \\
 & \text{with } x_i^* = d_{il}, & \\
(\nabla F(x^*))_i + \frac{df_{il}}{dx_i}(d_{il}) + \sum_{k=1}^{m}\sum_{j=0}^{M_i} a_{ij}^k\lambda_{ij}^k \ge 0, & \text{for all } i \in E(x^*), & \\
 & \text{with } x_i^* = d_{il}, & \\
\bar{u}_k = \sum_{j=0}^{M_i}\lambda_{ij}^k, \;\; i=1,\ldots,n, \;\; k=1,\ldots,m, & & \\
\bar{u}, \lambda \ge 0, & & \\
\lambda_{ij}^k((y^*)_i^k - a_{ij}^k x_i^* - b_{ij}^k) = 0, \;\; i=1,\ldots,n, & j=0,\ldots,M_i, & \\
 & k=1,\ldots,m, & \\
\bar{u}_k(\sum_{i=1}^{n}(y^*)_i^k - h^k) = 0, \;\; k=1,\ldots,m. & &
\end{array}\right\} \quad (3.7)$$

We next show how to derive (3.6) from (3.7). Consider an index $i \in N(x^*)$ with $x_i^* \in (d_{il}, d_{il+1})$. By the definition of $(y^*)_i^k$, $(y^*)_i^k = a_{il}^k x_i^* + b_{il}^k$. Then, by Lemma 3.1, the inequality

$$(y^*)_i^k \ge a_{ij}^k x_i^* + b_{ij}^k, \;\; j=0,\ldots,M_i$$

can be satisfied as equality only if $a_{ij}^k = a_{il}^k$. From the complimentary slackness, it follows that the dual variables λ_{ij}^k corresponding to the remaining values of j are all equal to zero. Therefore for each index $i \in N(x^*)$ with $x_i^* \in (d_{il}, d_{il+1})$ we have

$$\sum_{k=1}^{m}\sum_{j=0}^{M_i} a_{ij}^k\lambda_{ij}^k = \sum_{k=1}^{m} a_{il}^k \sum_{j=0}^{M_i}\lambda_{ij}^k. \quad (3.8)$$

Consider any $i \in E(x^*)$ with $x_i^* = d_{il}$. Again Lemma 3.1 implies that the inequality

$$(y^*)_i^k \ge a_{ij}^k x_i^* + b_{ij}^k$$

can be satisfied as equality only if $a_{ij}^k = a_{il}^k$ or $a_{ij}^k = a_{il-1}^k$. From the complimentary slackness, it follows that the dual variables λ_{ij}^k corresponding to the remaining values of j are all equal to zero.

Using the inequalities (3.4), we obtain that for every index $i \in E(x^*)$ with $x_i^* = d_{il}$

$$\sum_{k=1}^{m} a_{il-1}^k \sum_{j=0}^{M_i} \lambda_{ij}^k \leq \sum_{k=1}^{m} \sum_{j=0}^{M_i} a_{ij}^k \lambda_{ij}^k \leq \sum_{k=1}^{m} a_{il}^k \sum_{j=0}^{M_i} \lambda_{ij}^k. \tag{3.9}$$

Combining (3.8), (3.9) with the optimality conditions (3.7) yields

$$\left.\begin{array}{ll} (\nabla F(x^*))_i + \frac{df_{il}}{dx_i}(x_i^*) + \sum_{k=1}^m a_{il}^k \sum_{j=0}^{M_i} \lambda_{ij}^k = 0, & \text{for all } i \in N(x^*), \\ & \text{with } x_i^* \in (d_{il}, d_{il+1}), \\ (\nabla F(x^*))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il-1}^k \sum_{j=0}^{M_i} \lambda_{ij}^k \leq 0, & \text{for all } i \in E(x^*), \\ & \text{with } x_i^* = d_{il}, \\ (\nabla F(x^*))_i + \frac{df_{il}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il}^k \sum_{j=0}^{M_i} \lambda_{ij}^k \geq 0, & \text{for all } i \in E(x^*), \\ & \text{with } x_i^* = d_{il}. \end{array}\right\}$$

Since $\bar{u}_k = \sum_{j=0}^{M_i} \lambda_{ij}^k$ for every $i = 1, \ldots, n$, we can simplify the above conditions as follows.

$$\left.\begin{array}{ll} (\nabla F(x^*))_i + \frac{df_{il}}{dx_i}(x_i^*) + \sum_{k=1}^m a_{il}^k \bar{u}_k = 0, & \text{for all } i \in N(x^*), \\ & \text{with } x_i^* \in (d_{il}, d_{il+1}), \\ (\nabla F(x^*))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il-1}^k \bar{u}_k \leq 0, & \text{for all } i \in E(x^*), \\ & \text{with } x_i^* = d_{il}, \\ (\nabla F(x^*))_i + \frac{df_{il}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il}^k \bar{u}_k \geq 0, & \text{for all } i \in E(x^*), \\ & \text{with } x_i^* = d_{il}. \end{array}\right\} \tag{3.10}$$

Suppose that one of the piece-wise linear constraints of the problem (3.1) is satisfied by x^* as a strict inequality, i.e. $g^k(x^*) < h^k$ for some k. Then, by the definition of $(y^*)_i^k$,

$$\sum_{i=1}^{n} (y^*)_i^k < h^k,$$

and from the complimentary slackness $\bar{u}_k = 0$. Therefore

$$\bar{u}_k(g^k(x^*) - h^k) = 0,\ k = 1, \ldots, m. \tag{3.11}$$

Combining (3.10) and (3.11) with the feasibility of x^* and non-negativity of $\bar{u}$, we obtain that the vector $u^* = \bar{u}$ together with x^* satisfy (3.6). This proves the theorem. □

3.2 Algorithm 2

In this section, we formulate a solution algorithm for (3.1). The structure of Algorithm 2 is very similar to the structure of Algorithm 1. However, we have to modify the subproblem formed in Step 1, taking into account the piece-wise linear constraints. Recall that at the j-th iteration of Algorithm 1 we are looking for the next feasible point within the n-dimensional cube defined by D^j at this iteration. Instead of adding to the subproblem a piece-wise linear constraint, we add a linear constraint which coincides with this piece-wise linear constraint on the above mentioned cube. Therefore in Step 1.1, for each $k \in I^j$ we add an equality constraint

$$\sum_{i=1}^{n} (a^k_{i,D^j_i} x_i + b^k_{i,D^j_i}) = h^k.$$

In Step 3.2, the definition of the variables v^j_i and w^j_i is modified to reflect the piece-wise linear nature of the constraints and optimality conditions of Theorem 3.1.

Algorithm 2

Model Problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s. t.} \quad & g^k(x) \le h^k \ k = 1, \ldots, m, \end{aligned}$$

where $f(x)$ is defined by (2.2) and Assumption 2.1 is satisfied, $g^k(x)$ $k = 1, \ldots, m$ are defined by (3.2).

Initialization

Begin with any $x^0 \in S$.

For each $i = 1, \ldots, n$ define the working interval indicator $D_i^0 = l$ if $x_i^0 \in [d_{il}, d_{il+1})$.

Chose an initial working set of breakpoints E^0 such that $E^0 \subseteq E(x^0)$ and the vectors $\{(a_{1l}^k, \ldots, a_{nl}^k),\ k \in I^0, e_i,\ i \in E^0\}$ are linearly independent.

Set $j = 0$.

Step 1: Computation of Search Direction

Define the objective function f^j, gradients of the equality constraints α_k^j, $k = 1, \ldots, m$ and vector β^j for the j-th subproblem

$$\begin{aligned} & (\alpha_k^j)_i = a^k_{i,D_i^j}, \text{ for each } k = 1, \ldots, m,\ i = 1, \ldots, n, \\ & (\beta^j)_k = h^k - \textstyle\sum_{i=1}^n b^k_{i,D_i^j}, \text{ for each } k = 1, \ldots, m, \\ & f^j(x) = F(x) + \textstyle\sum_{i=1}^n f_{i,D_i^j}(x_i). \end{aligned}$$

Step 1.1

Consider the equality constrained minimization problem

$$\begin{aligned} \min \quad & f^j(x) \\ \text{s. t.} \quad & (\alpha_k^j)^T x = \beta_k^j,\ k \in I^j, \\ & x_i = d_{il},\ i \in E^j, \text{ with } l = D_i^j. \end{aligned} \tag{3.12}$$

If x^j is a stationary point for (3.12), for all $k \in I^j$ take u_k^j to be the Lagrange multiplier for the constraint $(\alpha_k^j)^T x = b^k$, for all $i \in E^j$ take λ_i^j to be the Lagrange multiplier for the constraint $x_i = d_{il}$ of (3.12), and proceed to Step 3.2.

Otherwise, find y^j such that it is feasible for (3.12) and $f^j(y^j) < f^j(x^j)$. Set $s^j = y^j - x^j$ and go to Step 2.

Step 2: Computation of the Step Size

Compute the smallest indices i_0, i_1 and i_2 such that

$$\sigma_0 = \frac{\beta_{i_0}^j - (\alpha_{i_0}^j)^T x^j}{(\alpha_{i_0}^j)^T s^j} = \min\left\{ \frac{\beta_k^j - (\alpha_k^j)^T x^j}{(\alpha_k^j)^T s^j} \ : \ k \notin I^j \text{ and } (\alpha_k^j)^T s^j > 0 \right\}, \quad (3.13)$$

$$\sigma_1 = \frac{d_{i_1 l} - x_{i_1}^j}{s_{i_1}^j} = \min\left\{ \frac{d_{il} - x_i^j}{s_i^j} \ : \ i \notin E^j \text{ and } s_i^j < 0, \ l = D_i^j \right\}, \quad (3.14)$$

$$\sigma_2 = \frac{d_{i_2 l+1} - x_{i_2}^j}{s_{i_2}^j} = \min\left\{ \frac{d_{il+1} - x_i^j}{s_i^j} \ : \ i \notin E^j \text{ and } s_i^j > 0, \ l = D_i^j \right\}. \quad (3.15)$$

If none of σ_0, σ_1 nor σ_2 were calculated, set $\bar{\sigma} = +\infty$, otherwise set

$$\bar{\sigma} = \min\{\sigma_0, \sigma_1, \sigma_2\}.$$

Perform a line search to minimize $f^j(x^j + \sigma s^j)$ over $0 \leq \sigma \leq \bar{\sigma}$. Let σ^j be the optimal step length for this problem.

If $\sigma^j = +\infty$, stop, the problem is unbounded from below. Otherwise, go to Step 3.1.

Step 3: Update

Step 3.1 Set

$$x^{j+1} = x^j + \sigma^j s^j.$$

If $\sigma^j < \bar{\sigma}$, set $I^{j+1} = I^j$, $E^{j+1} = E^j$, $D^{j+1} = D^j$, $j = j + 1$ and go to Step 1.

Otherwise

(i) If $\bar{\sigma} = \sigma_0$ set $I^{j+1} = I^j \cup \{i_0\}$, $E^j = E^j$, $D^{j+1} = D^j$, $j = j + 1$; go to Step 1.

(ii) If $\bar{\sigma} = \sigma_1$ set $I^{j+1} = I^j$, $E^{j+1} = E^j \cup \{i_1\}$, $D^{j+1} = D^j$, $j = j + 1$; go to Step 1.

(iii) If $\bar{\sigma} = \sigma_2$ set $I^{j+1} = I^j$, $E^{j+1} = E^j \cup \{i_2\}$ and $D^{j+1}_{i_2} = D^j_{i_2} + 1$, $D^{j+1}_i = D^j_i$ for all $i \neq i_2$, $j = j + 1$; go to Step 1.

Step 3.2 Set

$$x^{j+1} = x^j.$$

For each $i \in E^j$ calculate

$$v^j_i = -(\nabla F(x^j))_i - \frac{df_{il-1}}{dx_i}(x^j_i) - \sum_{k \in I^j} a^k_{il-1} u^j_k, \tag{3.16}$$

$$w^j_i = (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x^j_i) + \sum_{k \in I^j} a^k_{il} u^j_k, \tag{3.17}$$

where $l = D^j_i$.

If $u^j_k \geq 0$ for all $k \in I^j$, $v^j_i \geq 0$ and $w^j_i \geq 0$ for all $i \in E^j$, stop with optimal solution x^j.

Otherwise, calculate

$$u_{k_0} = \min_{k \in I^j}\{u^j_k\},$$
$$v_{k_1} = \min_{i \in E^j}\{v^j_i\},$$
$$w_{k_2} = \min_{i \in E^j}\{w^j_i\}.$$

Set

$$\bar{u} = \min\{u_{k_0}, v_{k_1}, w_{k_2}\}.$$

(i) If $\bar{u} = u_{k_0}$ set $I^{j+1} = I^j - \{k_0\}$, $E^{j+1} = E^j$, $D^{j+1} = D^j$, $j = j + 1$; go to Step 1.

(ii) If $\bar{u} = v_{k_1}$ set $I^{j+1} = I^j$, $E^{j+1} = E^j - \{k_1\}$ and $D^{j+1}_{k_1} = D^j_{k_1} - 1$, $D^{j+1}_i = D^j_i$ for all $i \neq k_1$, $j = j + 1$; go to Step 1.

(iii) If $\bar{u} = w_{k_2}$ set $I^{j+1} = I^j$, $E^{j+1} = E^j - \{k_2\}$, $D^{j+1} = D^j$, $j = j + 1$; go to Step 1.

3.3 Termination of Algorithm 2

In this section, we prove that Algorithm 2 indeed gives an optimal solution of the problem (3.1) or determines that this problem is unbounded. We use the notation introduced in the previous section.

Lemma 3.3 *Let x^j be the j-th iterate of Algorithm 2. Then*

(a) $E^j \subseteq E(x^j)$, for all $i = 1, \ldots, n$ $x^j_i \in [d_{il}, d_{il+1}]$ where $D^j_i = l$, furthermore if $i \in E^j$ then $x^j_i = d_{il}$.

(b) x^j is feasible for (3.1) and $I^j \subseteq I(x^j)$.

(c) $f(x^{j+1}) \leq f(x^j)$.

(d) $\{\alpha^j_k,\ k \in I^j, e_i,\ i \in E^j\}$ is a linearly independent set.

Proof. The above is true for x^0 by construction. Suppose (a)-(d) are also satisfied for x^j. Let us show that (a)-(d) are satisfied for x^{j+1}. There are two cases to be considered.

Case 1. Iterate x^{j+1} was obtained in Step 3.1. In this case

$$x^{j+1} = x^j + \sigma^j s^j.$$

(a) Chose any i and let $l = d_i^j$.

If $i \in E^j$, then $s_i^j = 0$ by construction. Therefore

$$x_i^{j+1} = d_{il}. \tag{3.18}$$

It follows form (3.18) that $E^j \subseteq E(x^{j+1})$.

Assume now that $i \notin E^j$. If $s_i^j < 0$ then using (3.14) and the fact that $\sigma^j \leq \sigma_1$, we obtain that

$$d_{il+1} \geq x_i^j \geq x^j + \sigma_i^j s_i^j \geq x^j + \sigma_1 s_i^j = d_{il}. \tag{3.19}$$

Similarly, if $s_i^j > 0$ then using (3.15) and the fact that $\sigma^j \leq \sigma_2$, we obtain that

$$d_{il} \leq x_i^j \leq x^j + \sigma_i^j s_i^j \leq x^j + \sigma_2 s_i^j = d_{il+1}. \tag{3.20}$$

Combining (3.18), (3.19) and (3.20), we obtain that for every i, $x_i^{j+1} \in [d_{il}, d_{il+1}]$. Thus (a) is true if $E^{j+1} = E^j$ and $D^{j+1} = D^j$ and it remains to consider the case when $E^{j+1} \neq E^j$ or $D^{j+1} \neq D^j$.

If $\sigma^j = \sigma_1$, then $E^{j+1} = E^j \cup \{i_1\}$. From the definition of σ_1, it follows that $x_{i_1}^j = d_{i_1 l}$, and therefore the index i_1 also belongs to $E(x^{j+1})$.

If $\sigma^j = \sigma_2$, then $E^{j+1} = E^j \cup \{i_2\}$ and $D_{i_2}^{j+1} = D_{i_2}^j + 1$. Again, from the definition of σ_2, it follows that $x_{i_2}^j = d_{i_2 l+1}$. It remains to note that $l + 1 = D_{i_2}^{j+1}$. Hence the index i_2 belongs to $E(x^{j+1})$, moreover $x_{i_2}^{j+1} = d_{i_2 l}$ where $l = D_{i_2}^{j+1} = D_{i_2}^j + 1$. This completes the prof of (a) for Case 1.

(b) If $k \in I^j$, then by construction $\alpha_k^j s^j = 0$. Therefore

$$\alpha_k^j x^{j+1} = \alpha_k^j x^j + \sigma^j \alpha_k^j s^j = \beta_k^j. \tag{3.21}$$

Assume now that $k \notin I^j$. If $(\alpha_k^j)^T s^j < 0$ then

$$\alpha_k^j x^{j+1} = \alpha_k^j x^j + \sigma^j \alpha_k^j s^j \leq \alpha_k^j x^j \leq \beta_k^j. \tag{3.22}$$

If $(\alpha_k^j)^T s^j > 0$, then combining (3.13) with the fact that $\sigma^j \leq \sigma_0$ we obtain

$$\alpha_k^j x^{j+1} = \alpha_k^j x^j + \sigma^j \alpha_k^j s^j \leq \alpha_k^j x^j + \sigma^0 \alpha_k^j s^j = \beta_k^j. \tag{3.23}$$

From (3.21), (3.22), and (3.23) it follows that

$$\alpha_k^j x^{j+1} \leq \beta_k^j, \ k = 1, \ldots, m. \tag{3.24}$$

Combining (a), (3.24), and Lemma 3.1, we conclude that x^{j+1} is feasible for (3.1). From (3.21), we obtain that

$$\alpha_k^j x^{j+1} = \beta_k^j, \ k \in I^j. \tag{3.25}$$

Again, combining (a), (3.25), and Lemma 3.1, we conclude that $I^j \subseteq I(x^{j+1})$.

It remains to consider the case when $I^j \neq I^{j+1}$. This can only happen if $\sigma^j = \sigma_0$ and $I^{j+1} = I^j \cup \{i_0\}$. It follows from the definition of σ_0, that the inequality (3.23) holds as equality in this case, and therefore the index i_0 belongs to $I(x^{j+1})$. Thus $I^{j+1} \subseteq I(x^{j+1})$.

(c) Since s^j was obtained by minimizing $f^j(x^j + \sigma s^j)$ over $0 \leq \sigma \leq \bar{\sigma}$, we have

$$f^j(x^j + \sigma s^j) \leq f^j(x^j + 0 s^j) = f^j(x^j).$$

By Lemma B.1 (Appendix B), $f(x)$ coincides with $f^j(x)$ for any x, such that $x_i^j \in [d_{il}, d_{il+1}]$, where $l = D_i^j$. Both x^j and x^{j+1} satisfy this conditions, implying that

$$f(x^{j+1}) \leq f(x^j).$$

(d) By construction, (d) holds true for the first iteration. Suppose that the vectors

$$\{\alpha_k^j,\ k \in I^j, e_i,\ i \in E^j\} \tag{3.26}$$

are linearly independent.

If Step 3.1. (i) was performed, then the index i_0 was added to the set I^j to form I^{j+1}. Assume that the set obtained by adding $\alpha_{i_0}^j$ to the vectors (3.26) is linearly dependent. Then we can find numbers $\zeta_k,\ k \in I^j, \eta_i,\ i \in E^j$, such that

$$\alpha_{i_0}^j = \sum_{k \in I^j} \zeta_k a^k + \sum_{i \in E^j} \eta_i e_i.$$

Taking the inner product of both sides of this equation with s^j and using the fact that $(\alpha_k^j)^T s^j = 0$ for each $k \in I^j$ and $(e_i)^T s^j = s_i^j = 0$ we obtain

$$(\alpha_{i_0}^j)^T s^j = \sum_{k \in I^j} \zeta_k (\alpha_k^j)^T s^j + \sum_{i \in E^j} \eta_i s_i^j = 0.$$

This is impossible in view of (2.9).

If Step 3.1 (ii) was performed, then the index i_1 was added to the set E^j to form E^{j+1}. Again, assume that the set obtained by adding $\alpha_{i_0}^j$ to the vectors (3.26) is linearly dependent. Then there exist $\zeta_k,\ k \in I^j,\ \eta_i,\ i \in E^j$, such that

$$e_{i_1} = \sum_{k \in I^j} \zeta_k a^k + \sum_{i \in E^j} \eta_i e_i.$$

Taking the inner product of both sides of this equation with s^j and using the fact that $(\alpha_k^j)^T s^j = 0$ for each $k \in I^j$ and $(e_i)^T s^j = s_i^j = 0$ we obtain

$$(e_{i_1})^T s^j = \sum_{k \in I^j} \zeta_k (\alpha_k^j)^T s^j + \sum_{i \in E^j} \eta_i s_i^j = 0.$$

Therefore $(e_{i_1})^T s^j = s_{i_1}^j = 0$ which contradicts (2.10).

Similarly, if Step 3.1 (iii) was performed, we conclude that adding e_{i_2} to the vectors (3.26) produces a linearly independent set.

Thus the vectors $\{\alpha_k^j,\ k \in I^{j+1},\ e_i,\ i \in E^{j+1}\}$ are linearly independent. Note that if Step 3.1(iii) was performed, then $\alpha_{i2}^j \neq \alpha_{i_2}^{j+1}$. Since $i_2 \in E^{j+1}$, it follows from Lemma C.1 (Appendix C) that the vectors $\{\alpha_k^{j+1},\ k \in I^{j+1}, e_i,\ i \in E^{j+1}\}$ are linearly independent as well.

Case 2: Iterate x^{j+1} was obtained in Step 3.2 and $x^{j+1} = x^j$. The feasibility and the fact that $f(x^{j+1}) = f(x^j)$ are obvious in this case.

By the assumption, $E^j \subseteq E(x^j) = E(x^{j+1})$, $I^j \subseteq I(x^j) = I(x^{j+1})$. Since at Step 3.2 we can only drop indices from the sets E^j and I^j, we also have $E^{j+1} \subseteq E(x^{j+1})$ and $I^{j+1} \subseteq I(x^{j+1})$. Thus in order to establish (a), (b), and (c) we only need to consider the case when $D^{j+1} \neq D^j$. This can only happen if $\bar{u} = v_{k_1}$ and $E^{j+1} = E^j - \{i_1\}$. Then $D_{k_1}^{j+1} = D_{k_1}^j - 1$ and $x_{k_1}^{j+1} = d_{k_1 l+1}$ where $l = D_{k_2}^{j+1} = D_{k_2}^j - 1$. So, k_2 no longer belongs to the working set of break points E^{j+1}, and $x_{k_1}^{j+1} \in [d_{k_1 l}, d_{k_1 l}]$ as required.

(d) Since one of the vectors in (3.26) is dropped the new vectors are still linearly independent. □

Let x^j be a stationary point for the j-th subproblem of Algorithm 2. Recall that the subproblem (3.12) is

$$\begin{aligned} \min \quad & f^j(x) \\ \text{s. t.} \quad & \alpha_k^j x = \beta_k^j,\ k \in I^j, \\ & x_i = d_{il},\ i \in E^j,\ \text{with } l = D_i^j. \end{aligned}$$

The optimality conditions for (3.12) are

$$\left.\begin{array}{lll} (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k\in I^j} a_{il}^k u_k = 0, & \text{for all } i \notin E^j, \\ & \text{where } l = D_i^j, \\ (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k\in I^j} a_{il}^k u_k + \lambda_i = 0, & \text{for all } i \in E^j, \\ & \text{where } l = D_i^j. \end{array}\right\} \tag{3.27}$$

We can define $u_k^j = 0$ for all k such that $k \notin I^j$. Note that this implies complimentary slackness.

The above optimality conditions for (3.12) can be written as

$$\left.\begin{array}{lll} (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il}^k u_k = 0, & \text{for all } i \notin E^j, \\ & \text{where } l = D_i^j, \\ (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il}^k u_k + \lambda_i = 0, & \text{for all } i \in E^j, \\ & \text{where } l = D_i^j. \end{array}\right\} \tag{3.28}$$

Note that for each $i \in E(x^j) - E^j$ with $x_i^j = d_{il}$ the conditions (3.28) imply that

$$(\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{iL}^k u_k = 0, \tag{3.29}$$

where $L \in \{l-1, l\}$.

Recall that by (3.4), $a_{il-1}^k \leq a_{il}^k$ for any fixed k and i, $k = 1, \ldots, m$, $i = 1, \ldots, n$. Since $u \geq 0$, we have

$$\sum_{k=1}^m a_{i-1}^k u_k \leq \sum_{k=1}^m a_{iL}^k u_k \leq \sum_{k=1}^m a_{il}^k u_k.$$

Thus x^j satisfies

$$\left.\begin{array}{ll}
(\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il}^k u_k^j = 0, & \text{for all } i \in N(x^j), \\
 & \text{with } x_i^j \in (d_{il}, d_{il+1}), \\
(\nabla F(x^j))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il-1}^k u_k^j \leq 0, & \text{for all } i \in E(x^j) - E^j, \\
 & \text{with } x_i^j = d_{il}, \\
(\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il}^k u_k^j \geq 0, & \text{for all } i \in E(x^j) - E^j, \\
 & \text{with } x_i^j = d_{il}.
\end{array}\right\} \quad (3.30)$$

Suppose now that Algorithm 2 terminates with x^j. This means that x^j is optimal for the equality constrained problem (3.12) and in addition, $u_k^j \geq 0$ for all $k \in I^j$, $v_i^j \geq 0$ and $w_i^j \geq 0$ for all $i \in E^j$.

Using the definition of v_i^j and w_i^j we obtain

$$\left.\begin{array}{lll}
(\nabla F(x^j))_i + \frac{df_{il-1}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il-1}^k u_k & \leq 0, & \text{for all } i \in E^j, \\
 & & \text{with } x_i = d_{il}, \\
(\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il}^k u_k & \geq 0, & \text{for all } i \in E^j, \\
 & & \text{with } x_i = d_{il}.
\end{array}\right\} \quad (3.31)$$

Conditions (3.30) and (3.31) together with the feasibility of x^j and complimentary slackness, imply that

$$\left.\begin{array}{ll}
g^k(x^j) \leq h^k, & k = 1, 2, \ldots, m, \\
(\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il}^k u_k^j = 0, & \text{for all } i \in N(x^j), \\
 & \text{with } x_i^j \in (d_{il}, d_{il+1}), \\
(\nabla F(x^j))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il-1}^k u_k^j \leq 0, & \text{for all } i \in E(x^j), \\
 & \text{with } x_i^j = d_{il}, \\
(\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il}^k u_k^j \geq 0, & \text{for all } i \in E(x^j), \\
 & \text{with } x_i^j = d_{il}, \\
u^j \geq 0, & \\
u_k^j (g^k(x^j) - h^k) = 0, & k = 1, \ldots, m.
\end{array}\right\} \quad (3.32)$$

If the above conditions are satisfied, then x^j is optimal for (3.1) by Theorem 3.1.

Suppose now that at the j-th iteration $\sigma^j = +\infty$. This can only happen if $\hat{\sigma} = \bar{\sigma} = +\infty$. Then for any $\sigma > 0$, $x^j + \sigma s^j$ is feasible and for very i, $(x_i^j + \sigma s_i^j) \in [d_{il}, d_{il+1}]$, where $l = D_i^j$ for $i = 1, \ldots, n$. Therefore $f(x^j + \sigma s^j) = f^j(x^j + \sigma s^j)$ on this halfline. In addition, f^j is unbounded from below on this halfline.

So, the objective function $f(x)$ is unbounded from below on a subset of the feasible set of (3.1) and therefore problem (3.1) is unbounded.

We summarize this in the following lemma.

Lemma 3.4

(i) Suppose that Algorithm 2 terminates with x^j. Then x^j is optimal for (3.1).

(ii) Suppose that Algorithm 2 determines that the problem is unbounded. Then (3.1) is indeed unbounded.

Suppose $x_1, \ldots, x_j, x_{j+1}, \ldots$ are the points generated by Algorithm 2. We established that

$$f(x^{j+1}) \leq f(x^j), \text{ for every } j = 0, 1, \ldots.$$

An important fact about an active set method is that under the non-degeneracy assumption the method can guarantee a strict decrease of the objective function from one iteration to the next at certain iterations. We next show how this can be generalized to our method. We start by recalling some results from differentiable optimization, and then show how these results apply in our case.

Consider the problem

$$\min\{f(x) \mid (a_i)^T x \leq b_i, i = 1, \ldots, m-1, (a_m)^T x = b_m\}, \tag{3.33}$$

where $f(x)$ is a twice differentiable convex function, $a_1, \ldots, a_m$ are n-vectors, and $b_1, \ldots, b_m$ are scalars. Suppose x^* is an optimal solution for (3.33). Let $u_1, \ldots, u_m$ be the corresponding dual multipliers.

Without the loss of generality we can assume that the first $k-1$ inequality constraints are active at x^*. Let

$$R = \{x \mid (a_i)^T x = b_i,\ i = 1, \ldots, k-1,\ (a_i)^T x \leq b_i, i = k, \ldots, m\}. \tag{3.34}$$

A point $x \in R^n$ is called **non-degenerate** for the problem (3.33) if the gradients of the constraints of this problem active at x are linearly independent.

Lemma 3.5 *Let R be defined by (3.34), x^* be a non-degenerate optimal solution for (3.33), u_m be a multiplier for constraint m, $u_m < 0$ and $(a_i)^T x^* = b_i$ for $i = 1, \ldots, k-1$. Then there exist a feasible descent direction s and a scalar $\bar{\sigma} > 0$ such that for any $\sigma < \bar{\sigma}$, $x^* + \sigma s \in R$ and*

$$f(x^* + \sigma s) < f(x^*).$$

See [6] for the proof.

We now show how Lemma 3.5 can be applied in the nondifferentiable case.

We start by generalizing the definition of a non-degenerate point.

Let x^j be an iterate of Algorithm 2 applied to problem (3.1). Assume that x^j is a stationary point for the subproblem from which it was obtained. Let α_k^j and β_k^j be defined by the j-th step of Algorithm 2. In order to apply the Lemma 3.5, we have to assume that the vectors

$$\{\alpha_k^j,\ k \in I(x^j),\ e_i,\ i \in E(x^j)\} \tag{3.35}$$

are linearly independent.

This motivates us to give the following definition.

A point $x \in R^n$ is called **non-degenerate** for the problem (3.1) if the vectors (3.35) are linearly independent.

This assumption seems very strong comparing to the standard non-degeneracy assumptions. This formulation is convenient to use in the proof, but it depends on the choice of a subproblem. In Appendix C we reformulate this assumption. We show that in fact it does not depend on the choice of the subproblem, roughly speaking, it is similar to the non-degeneracy of a corresponding point of the lifted problem.

Theorem 3.2 *Suppose that Algorithm 2 is applied to the problem (3.1). Let x^j be the j-th iterate of Algorithm 2. Assume that x^j is a non-degenerate point for the problem (3.1) and assume further that x^j is a stationary point for the subproblem number j. Suppose at least one of the parameters u_k^j, v_i^j or w_i^j defined by Step 3.2 of Algorithm 2 is negative. Then the iterate of Algorithm 2 x^{j+2} can be chosen, so that*

$$f(x^{j+2}) < f(x^j).$$

Proof. Let x^j be a stationary point of the j-th subproblem (3.12) of Algorithm 2. In the context of Algorithm 2, our next step would be to find the dual variables u_k^j corresponding to the constraints $a_k^j = b_k^j$ and calculate the parameters v_i^j and w_i^j, defined at x^j by (3.16), (3.17). Suppose that at least one of these numbers is negative. The algorithm proceeds by constructing the next subproblem in the following way:

Case 1. The dual variable $u_{k_0}^j$ corresponding to the equality constraint $a_{k_0}^j x = b_{k_0}^j$ is the smallest of the above numbers. Then the next subproblem is obtained from the j-th subproblem by dropping the constraint $a_{k_0}^j x = b_{k_0}^j$. $D_i^j = D_i^{j+1}$ for all

$i = 1, \ldots, n$. Therefore the objective function of the subproblem does not change. We can apply the result of Lemma 3.5 to the function $f^{j+1}(x)$ and

$$R = \left\{ x \;\middle|\; \begin{array}{ll} a_k^{j+1} x = b_k^{j+1}, & k \in I(x^j),\ k \neq k_0, \\ a_{k_0}^{j+1} x \leq b_{k_0}^{j+1}, & \\ x_i = d_{il}, & i \in E(x^j), \text{ with } l = D_i^{j+1} \end{array} \right\}. \tag{3.36}$$

Case 2.

$$v_{k_1}^j = -(\nabla F(x^{j+1}))_{k_1} - \frac{df_{k_1 l-1}}{dx_{k_1}}(x_{k_1}^{j+1}) - \sum_{k=1}^{m} a_{k_1 l-1}^k u_k < 0 \tag{3.37}$$

is the smallest of the numbers u_k^j, v_i^j, w_i^j.

There is a small technical difficulty in this case, since $f^{j+1}(x) \neq f^j(x)$.

We obtain the $j+1$-st subproblem from the j-th subproblem in two steps:

(i) Let $D_{k_1}^j = l$. Then we set $D_{k_1}^{j+1} = l - 1$. This means that in the objective function we substitute the term $f_{k_1 l}$ by $f_{k_1 l-1}$ and in each of the equality constraints we substitute the term $a_{k_1 l}^k x_{k_1} + b_{k_1 l}^k$ by $a_{k_1 l-1}^k x_{k_1} + b_{k_1 l-1}^k$. Since $x_{k_1}^j = d_{k_1 l}$, the above terms of the objective function and of the constraints coincide on x^j respectively. Therefore x^j is feasible for this intermediate problem. Moreover, combining all the equalities of (3.28), except for the equality number k_1, with the expression for $v_{k_1}^j$ we obtain

$$\left.\begin{array}{lll} (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il}^k u_k = 0, & \text{for all } i \notin E^j, & \\ & \text{with } x_i \in [d_{il}, d_{il+1}] & \\ (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il}^k u_k + \lambda_i = 0, & \text{for all } i \in E^j,\ i \neq k_1 & \\ & \text{with } x_i = d_{il}, & \\ (\nabla F(x^j))_i + \frac{df_{il-1}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il-1}^k u_k + v_{k_1}^j = 0, & \text{where } i = k_1 & \\ & \text{with } x_{k_1} = d_{il}. & \end{array}\right\}$$

This system implies that x^j is a stationary point for the intermediate subproblem, $v^j_{k_1} < 0$ is a dual multiplier corresponding to the constraint $x_{k_1} = d_{k_1 l}$.

(ii) We drop the constraint $x_{k_1} = d_{k_1 l}$ from the intermediate subproblem.

We can now apply Lemma 3.5 to the function $f^{j+1}(x)$ and

$$R = \left\{ x \;\middle|\; \begin{array}{ll} a^{j+1}_k x = b^{j+1}_k, & k \in I(x^j), \\ x_i = d_{il}, & i \in E(x^j),\ i \neq k_1,\ \text{with } l = D^{j+1}_i, \\ x_{k_1} \leq d_{k_1 l} & \text{with } l = D^{j+1}_{k_1} \end{array} \right\}. \tag{3.38}$$

Case 3.

$$w^j_{k_2} = -\lambda^j_{k_2} = (\nabla F(x^{j+1}))_{k_2} + \frac{df_{k_2 l}}{dx_{k_2}}(x^{j+1}_{k_2}) + \sum_{k=1}^{m} a^k_{k_2 l} u_k < 0 \tag{3.39}$$

is the smallest of the numbers u^j_k, v^j_i, w^j_i. We could say that $w^j_{k_2}$ is a dual variable corresponding to the equality constraint $-x_{k_2} = -d_{k_2 l}$. Then the next subproblem is obtained from the j-th subproblem by dropping the constraint $-x_{k_2} = -d_{k_2 l}$. We can apply the result of Lemma 3.5 to the function $f^{j+1}(x)$ and

$$R = \left\{ x \;\middle|\; \begin{array}{ll} a^{j+1}_k x = b^{j+1}_k, & k \in I(x^j), \\ x_i = d_{il}, & i \in E(x^j),\ i \neq k_2,\ \text{with } l = D^{j+1}_i, \\ x_{k_2} \geq d_{k_2 l} & \text{with } l = D^{j+1}_{k_2} \end{array} \right\}. \tag{3.40}$$

By Step 3.2, $x^{j+1} = x^j$. In each of the three cases, choosing the search direction s^{j+1} at x^{j+1}as described in Lemma 3.5 and small enough step size σ^{j+1} generates $x^{j+2} = x^{j+1} + \sigma^{j+1} s^{j+1}$ satisfying

$$f(x^{j+2}) < f(x^j).$$

□

Note that the subproblems of Algorithm 2 are convex linearly constrained minimization problems. In many cases such a subproblem can be solved in one step, or in a finite number of steps. In this case we can solve each subproblem of Algorithm 2 exactly.

Theorem 3.3 *Let Algorithm 2 be applied to the problem (3.1). Let $x^0 \in S$ be an initial point and let $x^1, x^2, ..., x^j, x^{j+1}, ...$ be the points obtained by the algorithm. Assume that*

(i) each subproblem of Algorithm 2 is solved in a finite number of steps;

(ii) if x^j is stationary for the subproblem from which it was obtained, then x^j is non-degenerate for the problem (3.1).

Then the optimal solution of the problem (3.1) will be determined in a finite number of steps or it will be concluded that the problem is unbounded from below.

Proof: Suppose we solve each subproblem exactly, and y^j is a solution. There are three cases to be considered:

(a) $x^j = y^j$, and thus x^j is a stationary point. In this case the update was performed by Step 3.2. If Algorithm 2 does not terminate at this point, by Theorem 3.2, we have $f(x^{j+2}) < f(x^j)$.

(b) $x^j \neq y^j$ and $\sigma^j < \bar{\sigma}$. This can only happen if $\sigma^j = 1$. Thus $x^{j+1} = x^j + s^j = y^j$. In this case, subproblem $j+1$ coincides with subproblem j, therefore x^{j+1} is a stationary point for the subproblem $j+1$.

(c) $x^j \neq y^j$ and $\sigma^j = \bar{\sigma}$. In this case, Step 3.1 applies and subproblem $j+1$ has one more constraint.

Note that after case (a) occurred, j-th subproblem will not be revisited. Case (b) subproblem is always followed by case (a) subproblem.

On the other hand, each time case (c) occurs, a constraint is added to the subproblem. Therefore case (a) or (b) should happen after at most n case (c) subproblems.

There are only finitely many subproblems that can be formed for this problem. Therefore Algorithm 2 will terminate in a finite number of steps. By Lemma 3.4, it either terminates with the optimal solution, or concludes that the problem is unbounded from below. □

3.4 Modified Algorithm 2

Note that we require that in Step 3.2 of Algorithm 2, the index corresponding to the smallest negative multiplier is removed form the corresponding working set, by analogy to the active set method. However this rule is not mandatory. It is sufficient to chose an index corresponding to any negative multiplier.

This suggests that we could formulate the last step of this algorithm in a slightly different way. Instead of choosing the minimum over all the indices, we can choose the minimum over a particular subset of indices. If every index in this subset is nonnegative, than we choose the minimum over the remaining indices. This technique, called **partial pricing**, is sometimes used in linear optimization [55].

We choose this subset in such a way, that the non-negativity of all the corresponding multipliers guarantees that the objective function is minimized by the current iterate x^j over some subset of the feasible region of the form

$$B^j = \left\{ x \;\middle|\; \begin{array}{ll} g^k(x) \le h^k, & k = 1, \ldots, m, \\ d_{il} \le x_i \le d_{il+1}, & i \notin E^j, \text{ where } l = D_i^j, \\ x_i = d_{il}, & i \in E^j, \text{where } l = D_i^j. \end{array} \right\} \tag{3.41}$$

After the algorithm minimizes the objective function over this region, next subset of the feasible region is chosen. The iterations at which the new region is chosen have a special status for this algorithm. We introduce index J to enumerate such iterations. We will denote the indices of these iterations by $j_0, j_1, \ldots, j_J$.

Let us show exactly how we are modifying Algorithm 2.

In the **Initialization** step let $J = 0$, $j_J = 0$, $j = 0$.

Reformulate Step 3.2 as follows:

Step 3.2

At the iteration number $j + 1$, such that $j_J \le j + 1$,

for each $i \in E^j$ calculate v_i^j and w_i^j as before ((3.16), (3.17)).

Form index sets L and U in the following way:

Start with $L = \emptyset$, $U = \emptyset$.

For each $i \in E^j - E^{j^J}$,

$$\text{if } x_i^j = d_{i,D_i^j} \text{ then } L = L \cup \{i\},$$

$$\text{else } U = U \cup \{i\}.$$

Calculate

$$u_{k_0} = \min_{k \in I^j} \{u_k^j\},$$
$$v_{k_1} = \min_{i \in U} \{v_i^j\},$$
$$w_{k_2} = \min_{i \in L} \{w_i^j\}.$$

Set

$$\hat{u} = \min\{u_{k_0}, v_{k_1}, w_{k_2}\}.$$

If $\hat{u} < 0$, set $\bar{u} = \hat{u}$. Go to Step 3.3.

Otherwise, if $\hat{u} \geq 0$, calculate

$$v_{k_1} = \min_{i \in L \cup E^{j_J}} \{v_i^j\},$$
$$w_{k_2} = \min_{i \in U \cup E^{j_J}} \{w_i^j\}.$$

Set

$$\bar{u} = \min\{v_{k_1}, w_{k_2}\}.$$

If $\bar{u} \geq 0$, stop with optimal solution x^j.

Otherwise, set $J = J + 1$, $j_J = j$. Go to Step 3.3.

Step 3.3

(i) If $\bar{u} = u_{k_0}$ set $I^{j+1} = I^j - \{k_0\}$, $E^{j+1} = E^j$, $D^{j+1} = D^j$, $j = j + 1$; go to Step 1.

(ii) If $\bar{u} = v_{k_1}$ set $I^{j+1} = I^j$, $E^{j+1} = E^j - \{k_1\}$ and $D_{k_1}^{j+1} = D_{k_1}^j - 1$, $D_i^{j+1} = D_i^j$ for all $i \neq k_1$, $j = j + 1$; go to Step 1.

(iii) If $\bar{u} = w_{k_2}$ set $I^{j+1} = I^j$, $E^{j+1} = E^j - \{k_2\}$, $D^{j+1} = D^j$, $j = j + 1$; go to Step 1.

This ends the description of the modified Step 3.2.

Note that for each iteration j, $j_J < j < j_{J+1}$, none of the equality constraints constraints corresponding to $i \in E^{j_J}$ can be dropped and $D_i^j = D_i^{j_J}$.

If an equality constraint is added by Step 3.1 (ii), index i_1 joins the set L. This constraint could only be dropped by Step 3.2 (iii) later on, but equality $D_{i_1}^j = D_{i_1}^{j_J}$ would still hold.

If an equality constraint is added by Step 3.1 (iii), index i_2 joins the set U and $D^j_{i_2} + 1 = D^{j_J}_{i_2}$. This constraint could only be dropped by Step 3.2.(ii), so that the interval indicator would be changed back to $D^j_{i_2} = D^{j_J}_i$.

Therefore,

$$D^j_i = D^{j_J}_i \text{ for each } i \notin U, \tag{3.42}$$

$$D^j_i = D^{j_J}_i + 1 \text{ for each } i \in U. \tag{3.43}$$

We now show how this modification allows us to simplify the formulation of Algorithm 2.

Lemma 3.6 *Let Step 3.2 of Algorithm 2 be modified as above. Let the modified algorithm be applied to the problem (3.1). Let $j = j_{J+1}$ for some $J > 0$. Then the iterate of the modified algorithm x^j minimizes the objective function of the problem (3.1) over a subset of the feasible region*

$$B^J = \left\{ x \;\middle|\; \begin{array}{lll} g^k(x) \le h^k, & k = 1, \ldots, m, & \\ d_{il} \le x_i \le d_{il+1}, & i \notin E^{j^J}, & \text{where } l = D^{j_J}_i, \\ x_i = d_{il}, & i \in E^{j^J}, & \text{where } l = D^{j_J}_i. \end{array} \right\} \tag{3.44}$$

Proof. By construction x^j is a stationary point for the j-th subproblem of the modified Algorithm 2. From the optimality conditions (3.30)

$$\left.\begin{array}{ll} (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x^j_i) + \sum_{k=1}^m a^k_{il} u^j_k = 0, & \text{for all } i \in N(x^j), \\ & \text{with } x^j_i \in (d_{il}, d_{il+1}), \\ (\nabla F(x^j))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + \sum_{k=1}^m a^k_{il-1} u^j_k \le 0, & \text{for all } i \in E(x^j) - E^j, \\ & \text{with } x^j_i = d_{il}, \\ (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(d_{il}) + \sum_{k=1}^m a^k_{il} u^j_k \ge 0, & \text{for all } i \in E(x^j) - E^j, \\ & \text{with } x^j_i = d_{il}. \end{array}\right\} \tag{3.45}$$

From the modified Step 3.2, $v_i^j \geq 0$ and $w_i^j \geq 0$ for every $i \in E^j - E^{j_J}$. By definition of v_i^j and w_i^j

$$\left.\begin{array}{lll} (\nabla F(x^j))_i + \frac{df_{il-1}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il-1}^k u_k \leq 0, & \text{for all } i \in U, & \\ & \text{with } x_i = d_{il}, & \\ (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il}^k u_k \geq 0, & \text{for all } i \in L, & \\ & \text{with } x_i = d_{il}. & \end{array}\right\} \tag{3.46}$$

Formulas (3.45) and (3.46) together with the feasibility of x^j and complimentary slackness, imply

$$\left.\begin{array}{ll} g^k(x^j) \leq h^k, & k = 1, \ldots, m, \\ (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(x_i^j) + \sum_{k=1}^m a_{il}^k u_k^j = 0, & \text{for all } i \in N(x^j), \\ & \text{with } x_i^j \in (d_{il}, d_{il+1}), \\ (\nabla F(x^j))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il-1}^k u_k^j \leq 0, & \text{for all } i \in E(x^j) - E^{j_J}, \\ & \text{with } x_i^j = d_{il},\ l = D_i^{j_J} + 1 \\ (\nabla F(x^j))_i + \frac{df_{il}}{dx_i}(d_{il}) + \sum_{k=1}^m a_{il}^k u_k^j \geq 0, & \text{for all } i \in E(x^j) - E^{j_J}, \\ & \text{with } x_i^j = d_{il},\ l = D_i^{j_J} \\ u^j \geq 0, & \\ u_k^j (g^k(x^j) - h^k) = 0, & k = 1, \ldots, m. \end{array}\right\} \tag{3.47}$$

By Lemma B.3 (Appendix B), x^j minimizes $f(x)$ over the set B^J. □

The modified Algorithm 2 solves the problem (3.1) by minimizing $f(x)$ over a sequence of subsets of the feasible region.

At the J-th iteration we find the solution of the J-th subproblem. If the subproblem is unbounded from below, the original problem is also unbounded from below. Otherwise we denote this solution by x^{J+1}. We check if x^{J+1} is optimal for (3.1). If not, we describe a procedure for constructing the next subproblem.

The big advantage of the modified algorithm is the possibility of easy implementation. If one has a software package available for solving the differentiable linearly constrained problems, one can fully benefit from the speed of this package. Note that consecutive subproblems differ by one constraint only, so one can use the optimal solution of the previous problem as a starting point for the next subproblem.

We give the detailed formulation of the algorithm below.

Modified Algorithm 2

Model Problem

$$\begin{aligned} \min \quad & f(x) \\ \text{s. t.} \quad & g^k(x) \leq h^k \ k = 1, \ldots, m, \end{aligned}$$

where $f(x)$ is defined by (2.2) and Assumption 2.1 is satisfied, $g^k(x)$ $k = 1, \ldots, m$ are defined by (3.2).

Initialization

Begin with any $x^0 \in S$.

Define the working interval indicator $D_i^0 = l$ if $x_i^0 \in [d_{il}, d_{il+1})$.

Define the initial working set of breakpoints E^0 such that $E^0 \subset E(x^0)$ and the vectors $\{a_{il},\ i \in I(x^0), e_i,\ i \in E^0\}$ are linearly independent.

Set $j = 0$.

Step 1: Solution of Subproblem Define the objective function f^j, gradients of

the equality constraints α_k^j, $k = 1, \ldots, m$ and vector b^j for the jth subproblem

$$
\begin{aligned}
&(\alpha_k^j)_i = a^k_{i,D_i^j}, \text{ for each } k = 1, \ldots, m, \ i = 1, \ldots, n, \\
&(\beta^j)_k = h^k - \textstyle\sum_{i=1}^n b^k_{i,D_i^j}, \text{ for each } k = 1, \ldots, m, \\
&f^j(x) = F(x) + \textstyle\sum_{i=1}^n f_{i,D_i^j}(x_i).
\end{aligned}
$$

Define the feasible set of the jth subproblem

$$
B^j = \left\{ x \;\middle|\; \begin{array}{rl} \alpha_k^j x \le \beta_k^j, & k = 1, \ldots, m, \\ d_{il} \le x_i \le d_{il+1}, & \text{for all } i \notin E^j, \text{ where } l = D_i^j, \\ x_i = d_{il}, & \text{for all } i \in E^j, \text{ where } l = D_i^j. \end{array} \right\}
$$

Solve the subproblem

$$
\min\{f^j(x) \mid x \in B^j\}. \tag{3.48}
$$

If the subproblem is unbounded from below, stop, the problem is unbounded. Otherwise denote the solution of this subproblem by x^{j+1}, denote the dual variables corresponding to the constraint $\alpha_k^j x \le \beta_k^j$ by u_k^{j+1}. Denote by E^{j+1} all the indices from $E(x^{j+1})$ that belong to the last working active set of this subproblem. Set $D_i^{j+1} = D_i^j$ for each $i = 1, \ldots, n$. Go to Step 2.

Step 2: Optimality Test

Form index sets L and U in the following way:

Start with $L = \emptyset$, $U = \emptyset$.

For each $i \in E^{j+1} - E^j$

$$
\text{if } x_i^j = d_{i,D_i^j} \text{ then } L = L \cup \{i\},
$$

$$
\text{else } U = U \cup \{i\},\ D_i^{j+1} = D_i^{j+1} + 1.
$$

For each $i \in L \cup E^j$, calculate

$$\begin{aligned} &v_i^{j+1} = -(\nabla F(x^{j+1}))_i - \tfrac{df_{il-1}}{dx_i}(x_i^{j+1}) - \textstyle\sum_{k=1}^m a_{il-1}^k u_k^{j+1}, \\ &\text{where } l = D_i^{j+1}. \end{aligned} \tag{3.49}$$

For each $i \in U \cup E^j$, calculate

$$\begin{aligned} &w_i^{j+1} = (\nabla F(x^{j+1}))_i + \tfrac{df_{il}}{dx_i}(x_i^{j+1}) + \textstyle\sum_{k=1}^m a_{il}^k u_k^{j+1}, \\ &\text{where } l = D_i^{j+1}. \end{aligned} \tag{3.50}$$

Compute

$$\begin{aligned} v_{k_1}^{j+1} &= \min_{i \in L \cup E^j} \{v_i^{j+1}\}, \\ w_{k_2}^{j+1} &= \min_{i \in U \cup E^j} \{w_i^{j+1}\}. \end{aligned}$$

If $v_{k_1}^{j+1}$, $w_{k_2}^{j+1} \geq 0$, or if no vs or ws are calculated, stop with optimal solution x^{j+1}. Otherwise go to Step 3.

Step 3: Constraint Modification

Case 1. If $v_{k_1}^{j+1} \leq w_{k_2}^{j+1}$,

set $E^{j+1} = E^{j+1} - \{k_1\}$, $j = j + 1$;

$D_{k_1}^{j+1} = D_{k_1}^{j+1} - 1$.

Case 2. If $v_{k_1}^{j+1} > w_{k_2}^{j+1}$,

set $E^{j+1} = E^{j+1} - \{k_2\}$, $j = j + 1$.

A similar algorithm was first proposed in [6] for the case when the transaction costs and constraints are linear. Their method was based on deriving the optimality conditions for the higher dimensional problem (1.6). We extended this algorithm to the case when the transaction costs are piece-wise linear and convex in [10] using

the optimality conditions (2.6). It was later shown in [7] that this algorithm can be also derived from the optimality conditions for the higher dimensional problem (1.6).

Example 3.1 *Consider (3.1) with*

$$F(x) = x_1^2 + x_2^2 - 2x_1 - 6x_2,$$

$$f_1(x) = \begin{cases} -0.2x & -0.2, \text{ if } x \leq -2, \\ -0.1x & , \text{ if } x \in [-2, 0], \\ 0.1x & , \text{ if } x \in [0, 2], \\ 0.2x & -0.2, \text{ if } x \geq 2, \end{cases}$$

and

$$f_2(x) = \begin{cases} -0.1x & -0.2, \text{ if } x \leq -2, \\ 0 & , \text{ if } x \in [-2, 0], \\ 0.1x & , \text{ if } x \in [0, 2], \\ 0.2x & -0.2, \text{ if } x \geq 2. \end{cases}$$

Suppose we add a constraint

$$|x_1| + |x_2| \leq 1.$$

This constraint can be rewritten as

$$g(x) \leq 1,$$

where $g(x) = g_1(x_1) + g_2(x_2)$ *and*

$$g_1(t) = g_2(t) = \begin{cases} -t & \text{if} \quad t \leq 0, \\ t & \text{if} \quad t > 0. \end{cases}$$

To be consistent with our notation, we rewrite the definition of $g_i(x_i)$ as

$$g_1(t) = g_2(t) = \begin{cases} -t & \text{if} \quad t \leq -2, \\ -t & \text{if} \quad t \in [-2, 0], \\ t & \text{if} \quad t \in [0, 2], \\ t & \text{if} \quad t > 2, \end{cases}$$

We solve the problem

$$\min\{f(x) \mid g(x) \leq 1\}.$$

Let $x^0 = (-0.5, -0.5)^T$ be a starting point.

Initialization.

$D_1^0 = 2$, $D_2^0 = 2$. $E^0 = \emptyset$.

Iteration 1.

Step 1.

$$f^0 = F(x) - 0.1x_1.$$

Solve the subproblem

$$\min\{f^0(x) \mid -2 \leq x_1 \leq 0,\ -2 \leq x_2 \leq 0,\ -x_1 - x_2 \leq 1\}.$$

The optimal solution of the subproblem is $x^1 = (0, 0)^T$, with $f(x^1) = 0$. Let $E^1 = \{1, 2\}$. $D_1^1 = 2$, $D_2^1 = 2$.

Step 2.

$L = \emptyset$, $U = \{1, 2\}$, $D_1^1 = 3$, $D_2^1 = 3$. Calculate $w^0 = (-1.9, -5.9)^T$. w_2^0 is the smallest of these numbers. $k_2 = 2$.

Step 3. Set $E^1 = \{1\}$.

Iteration 2.

$$f^1 = F(x) + 0.1x_1 + 0.1x_2.$$

Step 1. Solve the subproblem

$$\min\{f(x)^1 \mid x_1 = 0,\ 0 \leq x_2 \leq 2,\ x_1 + x_2 \leq 1\}.$$

The optimal solution of this subproblem is $x^2 = (0, 1)^T$, with $f(x^2) = -4.9$. Let $E^2 = \{1\}$. $D_1^2 = 3$, $D_2^2 = 3$.

Step 2. $L = \emptyset$, $U = \emptyset$.

Calculate $v_1^1 = 5.8 > 0$, $w_1^1 = 2 > 0$. Therefore x^2 is the optimal solution.

Note that the sequence of objective function values $f(x^0), f(x^1), f(x^2)$ is decreasing.

Figure 3.4 shows the feasible region of the example problem, subsets B^0 and B^1 defined by the algorithm, and the points, generated by the algorithm x^0, x^1, x^2.

3.5 Degeneracy

Let us first consider one natural example of a degenerate problem. Suppose the portfolio is rebalanced on a regular basis to meet the changes in the market data. In most cases the changes would not be very significant from one time period to the next one. Therefore we could expect that the change in the optimal portfolio

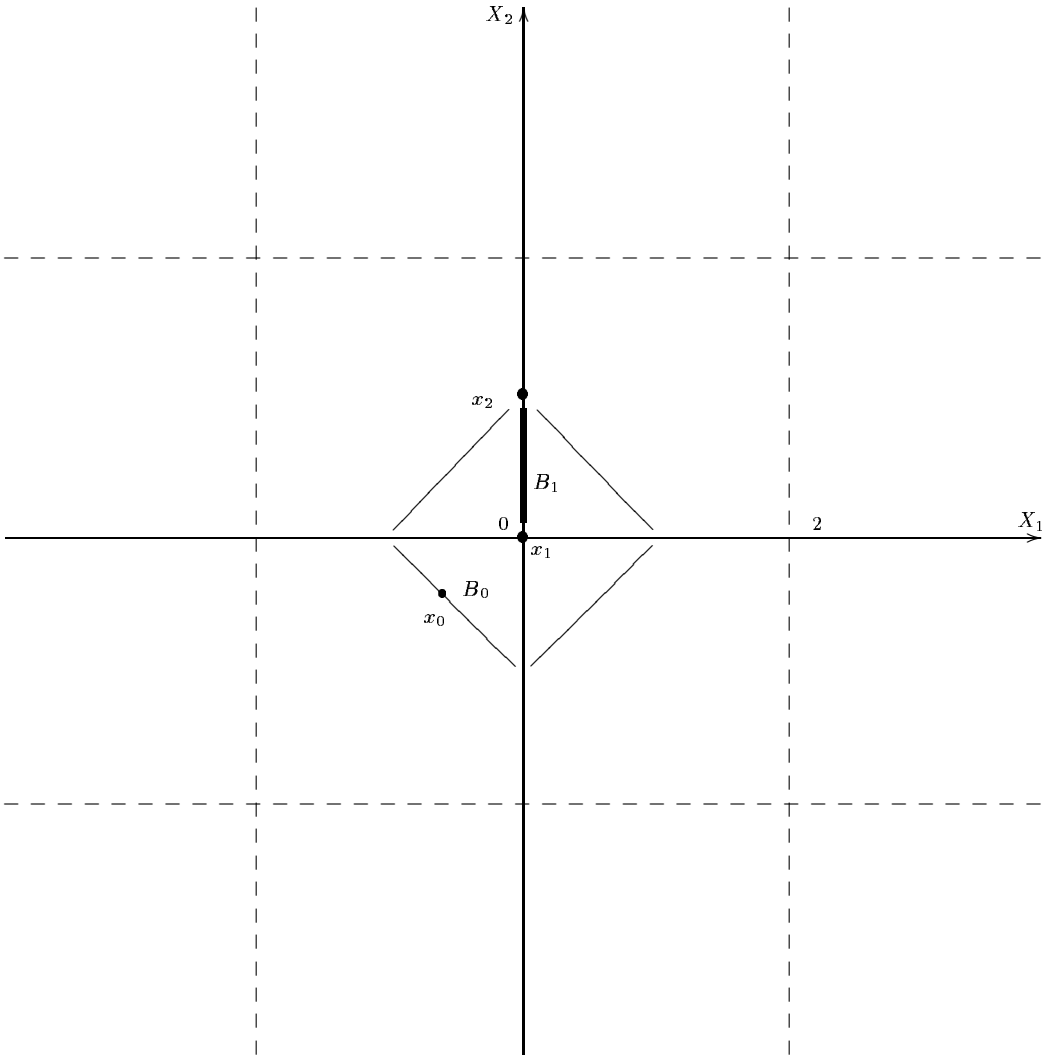

Figure 3.1: Progress of Modified Algorithm 2 in Solving Example 3.1.

is relatively small. It is natural to assume that the target portfolio $\hat{x}$, which was optimal for the previous time period is a good initial point for our optimization algorithm.

On the other hand, because of the structure of the transaction costs, at least n constraints (bounds) are active at $\hat{x}$. If at least one more constraint is active at the target portfolio, this point is degenerate.

In this section we suggest a way to resolve degeneracy issue in the Algorithm 2. Let x^j be the j-th iterate of the Algorithm 2. Suppose that x^j is degenerate.

In order to find a descent direction, we suggest solving an auxiliary piece-wise linear optimization problem, which approximates the original problem around x^j.

$$
\begin{array}{ll}
\min & \nabla F(x^j)z + \sum_{i=1}^{n} \frac{\partial f_i(x_i^j)}{\partial x_i} z_i \\
\text{s. t.} & \sum_{i \in N(x^j)} (a^k_{i,D_i(x^j)} z_i + b^k_{il}) + \quad \sum_{i \in E(x^j)} (g_i(z_i)) \leq h^k, \quad k = 1, \ldots, m, \\
& d_{il} \leq z_i \leq d_{il+1}, \quad i \in N(x^j), \ , l = D_i(x^j), \\
& d_{il-1} \leq z_i \leq d_{il+1}, \quad i \in E(x^j), \ , l = D_i(x^j).
\end{array}
$$

This problem could be converted into a linear optimization problem by introducing at most 2n new variables z^+ and z^- as described in the Appendix A.

It seems very likely that most of the standard techniques to deal with degeneracy, applied to the lifted $2n$ dimensional problem (1.6) can be modified to efficiently deal with degeneracy in our Algorithm.

3.6 Constraints on x^+, x^-

Let us recall the portfolio optimization problem with nondifferentiable convex transaction costs reformulated as a differentiable problem in a higher dimension (1.6).

$$\begin{aligned}
\min \quad & F(x) + \textstyle\sum_{i=1}^{n} \sum_{l=1}^{M_i^+} f_{il}^+(x_{il}^+) + \sum_{i=1}^{n} \sum_{l=1}^{M_i^-} f_{il}^-(x_{il}^-) \\
s.t. \quad & Ax \le b, \\
& x_i - \textstyle\sum_{i=1}^{n} \sum_{l=1}^{M_i^+} x_{il}^+ + \sum_{i=1}^{n} \sum_{l=1}^{M_i^-} x_{il}^- = \hat{x}_i, \text{ for } i = 1, ..., n, \\
& 0 \le x_{il}^+ \le d_{il}^+, \text{ for } i = 1, ..., n, \; l = 1, ..., M_i^+, \\
& 0 \le x_{il}^- \le d_{il}^-, \text{ for } i = 1, ..., n, \; l = 1, ..., M_i^-.
\end{aligned}$$

One of the advantages of this reformulation is that one can impose linear constraints of the type

$$G[x^+, x^-] \le h.$$

For example, we could add a turnover constraint as

$$\sum_{i=1}^{n} \sum_{l=1}^{M_i} x_i^{j+} \le K.$$

This constraint imposes an upper bound on the total amount bought.

Note that Algorithm 2 allows us to handle such constraints. But we make an assumption that the constraints should be convex. It appears at first sight that adding a constraint of type (3.51) to the problem (1.6) will allow us to handle nonconvex constraints. However, this is not the case.

First we note that one should be careful with adding such constraints. Let us recall how (1.6) was shown to be equivalent to the original problem (2.1). We argued that it is always best to set x_i^{1+} to the upper bound before taking $x_i^{2+} > 0$, and set x_i^{2+} to the upper bound before taking $x_i^{3+} > 0$ etc. If we add constraints

on x^+ and x^-, this argument might not be valid any more. However, this is not the case.

For example, addition of the constraint

$$x_i^{2+} = 0$$

could result in a solution with $x_i^{1+} > 0$, $x_i^{2+} = 0$ and $x_i^{3+} > 0$, which is meaningless for for our model. Another type of problem is that we could get a solution which forces us to buy and sell the same asset at the same time. This could occur if we have a lower bound on one of the x_i^{j+} or x_i^{j-}.

We would like to find a condition on the coefficients of the constraints, which would guarantee that the problem (1.6) with the additional constraints on x^+ and x^- is equivalent to the original problem.

Let us form a vector (x, x^m, x^p) form x in the following way[1]:

for each index i such that $x_i \geq \hat{x}_i$ and $x_i - \hat{x}_i \in [d_{iL}^+, d_{iL+1}^+]$

$$\begin{aligned}
&x_i^{lm} = 0, && \text{for} \quad l = 1, .., M_i^-, \\
&x_i^{lp} = d_{il}^+, && \text{for} \quad l = 1, .., L-1, \\
&x_i^{Lp} = x_i - \hat{x}_i - d_{iL}^+, && \\
&x_i^{lp} = 0, && \text{for} \quad l = L+1, .., M_i^+,
\end{aligned}$$

and for each index i such that $x_i < \hat{x}_i$ and $\hat{x}_i - x_i \in [d_{iL}^-, d_{iL+1}^-]$

$$\begin{aligned}
&x_i^{lm} = 0, && \text{for} \quad l = L+1, .., M_i^- \\
&x_i^{Lm} = \hat{x}_i - x_i - d_{iL}^-, && \\
&x_i^{lm} = d_{il}^-, && \text{for} \quad l = 1, .., L-1, \\
&x_i^{lp} = 0, && \text{for} \quad l = 1, .., M_i^+.
\end{aligned}$$

[1]We wrote (x, x^m, x^p) instead of righting $(x^T, (x^m)^T, (x^p)^T)^T$. In what follows we will often interchange the row and column notation. The type of vector we are dealing with should be clear from the context.

We'll say that the vector (x, x^m, x^p) is formed from x in a **natural way**.

Let us consider a feasible region of (1.6) with an additional linear inequality constraint on x^+ and x^-:

$$\begin{aligned}
&Ax \le b,\\
&x_i - \textstyle\sum_{i=1}^{n}\sum_{l=1}^{M_i^+} x_{il}^+ + \sum_{i=1}^{n}\sum_{l=1}^{M_i^-} x_{il}^- = \hat{x}_i, \text{ for } i = 1, ..., n,\\
&0 \le x_{il}^+ \le d_{il}^+, \text{ for } i = 1, ..., n,\ l = 1, ..., M_i^+,\\
&0 \le x_{il}^- \le d_{il}^-, \text{ for } i = 1, ..., n,\ l = 1, ..., M_i^-\\
&\textstyle\sum_{i=1}^{n}\sum_{l=1}^{M_i^+} g_{il}^+ x_{il}^+ + \sum_{i=1}^{n}\sum_{l=1}^{M_i^-} g_{il}^- x_{il}^- \le h.
\end{aligned} \tag{3.51}$$

Assumption 3.1 *Suppose that for any feasible vector* (x, x^-, x^+) *, vector* (x, x^m, x^p) *formed from* x *in a natural way is also feasible.*

Under this assumption problem (1.6) is equivalent to the original portfolio problem. This condition would be difficult to check in general. Below we formulate a different condition, which is sufficient for the Assumption 3.1 to hold.

Lemma 3.7 *Let region* S *be defined by (3.51) and let*

$$-g_{iM_i^-}^- \le \cdots \le -g_{il}^- \le \cdots \le -g_{i1}^- \le g_{i1}^+ \le \cdots \le g_{il}^+ \le \cdots \le g_{iM_i+}^+.$$

Then for any vector (x, x^-, x^+) *that is feasible for (3.51), the corresponding vector* (x, x^m, x^p) *is also feasible for (3.51).*

Proof Case 1. Suppose $x_{ik}^+ < d_{ik+1}^+$ and $x_{ij}^+ > 0$ for some $k < j$. Then we can shift $\epsilon = \min\{d_{ik+1}^+ - x_{ik}^+, x_{ij}^+\}$ from x_{ik}^+ to x_{ij}^+ and the resulting vector will remain

feasible. Since $g_{ik}^{+} \leq g_{ij}^{+}$,

$$\sum_{i=1}^{n} \sum_{l=1, l \neq j,k}^{M_i^+} g_{il}^{+} x_{il}^{+} + \sum_{i=1}^{n} \sum_{l=1}^{M_i^-} g_{il}^{-} x_{il}^{-} + g_{ik}^{+}(x_{ik}^{+} + \epsilon) + g_{ij}^{+}(x_{ij}^{+} - \epsilon) =$$

$$\sum_{i=1}^{n} \sum_{l=1, l \neq j,k}^{M_i^+} g_{il}^{+} x_{il}^{+} + \sum_{i=1}^{n} \sum_{l=1}^{M_i^-} g_{il}^{-} x_{il}^{-} + g_{ik}^{+} x_{ik}^{+} + g_{ij}^{+} x_{ij}^{+} + \epsilon(g_{ik}^{+} - g_{ij}^{+}) \leq$$

$$\sum_{i=1}^{n} \sum_{l=1}^{M_i^+} g_{il}^{+} x_{il}^{+} + \sum_{i=1}^{n} \sum_{l=1}^{M_i^-} g_{il}^{-} x_{il}^{-} \leq h.$$

After a finite number of such steps Case 1 will no longer occur.

Case 2. $x_{ik}^{-} < d_{ik+1}^{-}$ and $x_{ij}^{-} > 0$ for some $k < j$. Then we can shift $\epsilon = \min\{d_{ik+1}^{-} - x_{ik}^{-}, x_{ij}^{-}\}$ from x_{ik}^{-} to x_{ij}^{-} and the resulting vector will remain feasible. Since $-g_{ij}^{-} \leq -g_{ik}^{-}$ or $g_{ik}^{-} \leq g_{ij}^{-}$,

$$\sum_{i=1}^{n} \sum_{l=1}^{M_i^+} g_{il}^{+} x_{il}^{+} + \sum_{i=1}^{n} \sum_{l=1, l \neq j,k}^{M_i^-} g_{il}^{-} x_{il}^{-} + g_{ik}^{-}(x_{ik}^{-} + \epsilon) + g_{ij}^{-}(x_{ij}^{-} - \epsilon) =$$

$$\sum_{i=1}^{n} \sum_{l=1}^{M_i^+} g_{il}^{+} x_{il}^{+} + \sum_{i=1}^{n} \sum_{l=1, l \neq j,k}^{M_i^-} g_{il}^{-} x_{il}^{-} + g_{ik}^{-} x_{ik}^{-} + g_{ij}^{-} x_{ij}^{-} + \epsilon(g_{ik}^{-} - g_{ij}^{-}) \leq$$

$$\sum_{i=1}^{n} \sum_{l=1}^{M_i^+} g_{il}^{+} x_{il}^{+} + \sum_{i=1}^{n} \sum_{l=1}^{M_i^-} g_{il}^{-} x_{il}^{-} \leq h.$$

After a finite number of such steps Case 2 will no longer occur.

Case 3. $x_{ik}^{+} > 0$ and $x_{ij}^{-} > 0$ for some k and j. Then we can deduct $\epsilon = \min\{x_{ik}^{+}, x_{ij}^{-}\}$ from both x_{ik}^{+} and x_{ij}^{-} and the resulting vector will remain feasible.

Since $-g_{ij}^- \leq g_{ik}^+$,

$$\sum_{i=1}^{n} \sum_{l=1, l\neq k}^{M_i^+} g_{il}^+ x_{il}^+ + \sum_{i=1}^{n} \sum_{l=1, l\neq j}^{M_i^-} g_{il}^- x_{il}^- + g_{ik}^+ (x_{ik}^+ - \epsilon) + g_{ij}^- (x_{ij}^- - \epsilon) =$$

$$\sum_{i=1}^{n} \sum_{l=1, l\neq k}^{M_i^+} g_{il}^+ x_{il}^+ + \sum_{i=1}^{n} \sum_{l=1, l\neq j}^{M_i^-} g_{il}^- x_{il}^- + g_{ik}^+ x_{ik}^+ + g_{ij}^- x_{ij}^- - \epsilon(g_{ik}^+ + g_{ij}^-) \leq$$

$$\sum_{i=1}^{n} \sum_{l=1}^{M_i^+} g_{il}^+ x_{il}^+ + \sum_{i=1}^{n} \sum_{l=1}^{M_i^-} g_{il}^- x_{il}^- \leq h.$$

After a finite number of such steps Case 3 will no longer occur.

If none of the above 3 cases apply, the resulting vector is the one obtained from x in a natural way and it is feasible.

□

Chapter 4

Interior-point Method

4.1 Smoothing via Splines

We would like to see if an interior point method (IPM) can be applied to solve problem (2.1) efficiently. Throughout this chapter we will assume that

$$F(x) = \frac{1}{2}x^T Gx + c^T x \tag{4.1}$$

is a strictly convex quadratic function on $\mathbb{R}^n$, G is symmetric positive semidefinite, and $f_i(x_i)$ is a piecewise linear function on $\mathbb{R}$, with break-points at d_{ik}, i.e.

$$f_i(x_i) = \begin{cases} f_{i0} := p_{i0}x_i + h_{i0}, & \text{if} \quad x_i \leq d_{i1}, \\ f_{il} := p_{il}x_i + h_{il}, & \text{if} \quad d_{il} \leq x_i \leq d_{il+1},\ l = 1, \ldots, M_i, \\ f_{iM_i} := p_{iM_i}x_i + h_{iM_i}, & \text{if} \quad x_i \geq d_{iM_i}. \end{cases} \tag{4.2}$$

Applying the IPM directly to the nondifferentiable problem would force us to follow a nondifferentiable "central path".

A point $x \in \mathbb{R}^n$ is a point on the "central path", if and only if, for $\mu > 0$,

$$\left.\begin{array}{ll}
Ax + s = b,\ s \geq 0 & \\
(\nabla F(x))_i + \frac{df_{il}}{dx_i}(x_i) + (A^T u)_i = 0, & \text{for all } i \in N(x), \\
 & \text{with } x_i \in (d_{il}, d_{il+1}), \\
(\nabla F(x))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + (A^T u)_i + v_i = 0, & \text{for all } i \in E(x), \\
 & \text{with } x_i = d_{il}, \\
0 \leq v_i \leq \frac{df_{il}}{dx_i}(d_{il}) - \frac{df_{il-1}}{dx_i}(d_{il}), & \text{for all } i \in E(x), \\
 & \text{with } x_i = d_{il}, \\
u \geq 0, & \\
u_i s_i = \mu,\ i = 1, \ldots, m, &
\end{array}\right\} \tag{4.3}$$

or

$$\left.\begin{array}{ll}
Ax + s = b,\ s \geq 0 & \\
(Gx)_i + c_i + p_{il} + (A^T u)_i = 0, & \text{for all } i \in N(x), \\
 & \text{with } x_i \in (d_{il}, d_{il+1}), \\
(Gx)_i + c_i + p_{il-1} + (A^T u)_i + v_i = 0, & \text{for all } i \in E(x), \\
 & \text{with } x_i = d_{il}, \\
0 \leq v_i \leq p_{il} - p_{il-1}, & \text{for all } i \in E(x), \\
 & \text{with } x_i = d_{il}, \\
u \geq 0, & \\
u_i s_i = \mu,\ i = 1, \ldots, m. &
\end{array}\right\} \tag{4.4}$$

Approximating the nondifferentiable functions $f_i(x_i)$ by smooth functions allows us to fully use the theory of differentiable optimization, and in particular, interior point methods.

There are two approaches that could be taken here. In many cases the nondifferentiable function $f_i(x_i)$ is just an approximation of some smooth function based

on a given data set. Then taking a convex cubic spline on this data set would give a better approximation of the original function and the objective of the optimization problem (2.1) would become smooth. For more details on this approach, see for example [47], and for a general reference on splines see [11].

However, in real life, the original data set is often not available and it is best to find a solution to the given data. Transaction cost functions, for example, are always non-smooth. Therefore, we focus on the second approach. We use smooth convex splines that approximate the given piecewise linear convex functions $f_i(x_i)$ that represent the transaction costs.

4.2 Interior Point Method for Smooth Approximations

Suppose the functions $f_i(x_i)$ are approximated by smooth functions $\bar{f}_i(x_i)$. We denote the first and second derivatives by $\bar{f}_i'(x_i)$ and $\bar{f}_i''(x_i)$, respectively.

Let (x, u, s) be a current iterate, with $(u, s) > 0$. The following system of perturbed optimality conditions will be considered at each iteration of the IPM

$$\left.\begin{array}{l} Ax + s = b,\ s > 0 \\ (Gx)_i + c_i + \bar{f}_i'(x_i) + (A^T u)_i = 0, \quad \text{for all } i = 1, \ldots, n, \\ u > 0, \\ u_i s_i = \mu,\ i = 1, \ldots, m, \end{array}\right\} \tag{4.5}$$

Newton's method is applied to this system.

Define the barrier parameter $\mu = \frac{u^T s}{m}$, the vector of first derivatives $g = (\bar{f}_1'(x_1), \ldots, \bar{f}_n'(x_n))^T$, and the diagonal matrices

$$U = \text{Diag}(u_1, \ldots, u_m), \quad S = \text{Diag}(s_1, \ldots, s_m), \quad H = \text{Diag}(\bar{f}_1''(x_1), \ldots, \bar{f}_n''(x_n)).$$

Then the search direction for (4.5) is found from the linearized system (Newton's equation)

$$\begin{bmatrix} G+H & A^T & 0 \\ A & 0 & I \\ 0 & S & U \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta u \\ \Delta s \end{pmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ -Us + \sigma\mu e \end{bmatrix}, \tag{4.6}$$

where the residuals

$$r_c = Gx + c + g + A^T u, \quad r_b = Ax + s - b,$$

e is the vector of ones, and $\sigma \in [0, 1]$ is the centering parameter.

We can use block eliminations to simplify the linearized system. We first solve

$$\Delta s = -U^{-1} S \Delta u - s + \sigma\mu U^{-1} e.$$

Then we can eliminate Δs and rewrite (4.6) as the symmetric, indefinite, linear system ($n + m$ sized, augmented or quasidefinite system)

$$\begin{bmatrix} G+H & A^T \\ A & -U^{-1}S \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta u \end{pmatrix} = \begin{bmatrix} -r_c \\ -r_b + s - \sigma\mu U^{-1} e \end{bmatrix}. \tag{4.7}$$

Further, since

$$\Delta u = S^{-1} U [A\Delta x + r_b - s + \sigma\mu U^{-1} e] = -u + S^{-1}[U(A\Delta x + r_b) + \sigma\mu e],$$

we can eliminate Δu and obtain the (n sized, normal equation system)

$$\begin{aligned} [G + H + A^T (S^{-1}U) A] \Delta x &= -r_c + A^T (S^{-1}U)[-r_b + s - \sigma\mu U^{-1} e] \\ &= -r_c + A^T [u - S^{-1}(U r_b + \sigma\mu e)] \\ &= -(Gx + c + g) - A^T S^{-1} [U r_b + \sigma\mu e]. \end{aligned} \tag{4.8}$$

We can add upper and lower bounds $b_l \leq x \leq b_u$ to the problem. Let $0 < (x, u, u_l, u_u, s, s_l, s_u)$ be a current iterate of the IPM , where u_l, u_u and s_l, s_u are the

dual variables and slack variables corresponding to the upper and lower bounds, respectively. In addition, we redefine the barrier parameter $\mu = \frac{u^T s + u_l^T s_l + u_u^T s_u}{m+2*n}$, and define the diagonal matrices

$$U_l = \text{Diag}(u_l),\ U_u = \text{Diag}(u_u),\ S_l = \text{Diag}(s_l),\ S_u = \text{Diag}(s_u).$$

Then the search direction is now found from the Newton equation

$$\begin{bmatrix} G+H & A^T & I & -I \\ A & -U^{-1}S & 0 & \\ I & 0 & -U_u^{-1}S_u & 0 \\ -I & 0 & 0 & -U_l^{-1}S_l \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta u \\ \Delta u_u \\ \Delta u_l \end{pmatrix} = \begin{bmatrix} -r_c \\ -r_b + s - \sigma\mu U^{-1}e \\ -r_{bu} + s_u - \sigma\mu U_u^{-1}e \\ -r_{bl} + s_l - \sigma\mu U_l^{-1}e \end{bmatrix}, \tag{4.9}$$

with residuals $r_c = Gx + c + g + A^T u$, $r_b = Ax + s - b$, $r_{bu} = x - b_u$, $r_{bl} = -x + b_l$. We can repeat the block eliminations and find

$$\begin{aligned}
\Delta s &= -U^{-1}S\Delta u - s + \sigma\mu U^{-1}e, \\
\Delta s_u &= -U_u^{-1}S_u\Delta u_u - s_u + \sigma\mu U_u^{-1}e, \\
\Delta s_l &= -U_l^{-1}S_l\Delta u_l - s_l + \sigma\mu U_l^{-1}e, \\
\Delta u_u &= S_u^{-1}U_u[\Delta x + r_{bu} - s_u + \sigma\mu U_u^{-1}e] = -u_u + S_u^{-1}[U_u(\Delta x + r_{bu}) + \sigma\mu e], \\
\Delta u_l &= S_l^{-1}U_l[-\Delta x + r_{bl} - s_l + \sigma\mu U_l^{-1}e] = -u_l + S_l^{-1}[U_l(-\Delta x + r_{bl}) + \sigma\mu e], \\
\Delta u &= S^{-1}U[A\Delta x + r_b - s + \sigma\mu U^{-1}e] = -u + S^{-1}[U(A\Delta x + r_b) + \sigma\mu e].
\end{aligned}$$

The linearized systems become:

$$\begin{bmatrix} G + H + U_u^{-1}S_u + U_l^{-1}S_l & A^T \\ A & -U^{-1}S \end{bmatrix} \begin{pmatrix} \Delta x \\ \Delta u \end{pmatrix} = \begin{bmatrix} r_0 \\ -r_b + s - \sigma\mu U^{-1}e \end{bmatrix}, \tag{4.10}$$

where

$$r_0 = -r_c + u_u - S_u^{-1}[U_u r_{bu} + \sigma\mu e] + u_l - S_l^{-1}[U_l r_{bl} + \sigma\mu e],$$

and

$$\begin{aligned}[G + H + U_u^{-1}S_u + U_l^{-1}S_l + A^T(S^{-1}U)A]\Delta x &= -r_0 + A^T(S^{-1}U)[-r_b + s - \sigma\mu U^{-1}e] \\ &= -r_0 + A^T[u - S^{-1}(Ur_b + \sigma\mu e)].\end{aligned} \tag{4.11}$$

4.3 Quadratic and Cubic Splines

Recall that $f_i(x_i)$ is a piecewise linear convex function. We can approximate it by a spline $\bar{f}_i(x_i, \epsilon)$ in the following way. Let

$$\bar{f}_i(x_i, \epsilon) = f_i(x_i) + s_i(x_i, \epsilon). \tag{4.12}$$

Let us denote $\Delta p_{il} = p_{il} - p_{il-1}$.

For the quadratic spline

$$s_i(x_i, \epsilon) = \begin{cases} \frac{\Delta p_{il}}{4\epsilon}(x_i - d_{il} + \epsilon)^2, & \text{if } x_i \in [d_{il} - \epsilon, d_{il} + \epsilon] \text{ for some } l \in \{1, \ldots, M_i\}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.13}$$

the first partial derivative of $s_i(x_i, \epsilon)$ with respect to x_i is

$$\frac{\partial s_i(x_i, \epsilon)}{\partial x_i} = \begin{cases} \frac{\Delta p_{il}}{2\epsilon}(x_i - d_{il} + \epsilon), & \text{if } x_i \in [d_{il} - \epsilon, d_{il} + \epsilon] \text{ for some } l \in \{1, \ldots, M_i\}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.14}$$

and the second partial derivative of $s_i(x_i, \epsilon)$ with respect to x_i is

$$\frac{\partial^2 s_i(x_i, \epsilon)}{\partial x_i^2} = \begin{cases} \frac{\Delta p_{il}}{2\epsilon}, & \text{if } x_i \in [d_{il} - \epsilon, d_{il} + \epsilon] \text{ for some } l \in \{1, \ldots, M_i\}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.15}$$

For the cubic spline

$$s_i(x_i) = \begin{cases} \frac{\Delta p_{il}}{6\epsilon^2}(x_i - d_{il} + \epsilon)^3, & \text{if } x_i \in [d_{il} - \epsilon, d_{il}] \text{ for some } l \in \{1, \ldots, M_i\}, \\ -\frac{\Delta p_{il}}{6\epsilon^2}(x_i - d_{il} - \epsilon)^3 + (\Delta p_{il})x_i, & \text{if } x_i \in [d_{il}, d_{il} + \epsilon] \text{ for some } l \in \{1, \ldots, M_i\}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.16}$$

the first partial derivative of $s_i(x_i, \epsilon)$ with respect to x_i is

$$\frac{\partial s_i(x_i, \epsilon)}{\partial x_i} = \begin{cases} \frac{\Delta p_{il}}{2\epsilon^2}(x_i - d_{il} + \epsilon)^2, & \text{if } x_i \in [d_{il} - \epsilon, d_{il}] \text{ for some } l \in \{1, \ldots, M_i\}, \\ -\frac{\Delta p_{il}}{2\epsilon^2}(x_i - d_{il} - \epsilon)^2 + \Delta p_{il}, & \text{if } x_i \in [d_{il}, d_{il} + \epsilon] \text{ for some } l \in \{1, \ldots, M_i\}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.17}$$

and the second partial derivative of $s_i(x_i, \epsilon)$ with respect to x_i is

$$\frac{\partial^2 s_i(x_i, \epsilon)}{\partial x_i^2} = \begin{cases} \frac{\Delta p_{il}}{\epsilon^2}(x_i - d_{il} + \epsilon), & \text{if } x_i \in [d_{il} - \epsilon, d_{il}] \text{ for some } l \in \{1, \ldots, M_i\}, \\ -\frac{\Delta p_{il}}{\epsilon^2}(x_i - d_{il} - \epsilon), & \text{if } x_i \in [d_{il}, d_{il} + \epsilon] \text{ for some } l \in \{1, \ldots, M_i\}, \\ 0 & \text{otherwise,} \end{cases} \tag{4.18}$$

It is trivial to check that the functions $\bar{f}_i(x_i, \epsilon)$ defined above are continuously differentiable. In case of the cubic spline, they are twice continuously differentiable.

Note that the largest value of ϵ that we want to use should satisfy

$$\epsilon < \frac{1}{2} \min_{i,l}(d_{il} - d_{il-1}).$$

Let us define

$$\bar{\epsilon} = \frac{1}{3} \min_{i,l}(d_{il} - d_{il-1}). \tag{4.19}$$

4.4 Sensitivity Analysis

Let us recall some basic sensitivity results, see [23].

Definition 4.1 *Let Γ be a point-to-set mapping from $T \subset \mathbb{R}^n$ to subsets of $\mathbb{R}^m$ and let $t_n \in T$ be such that $t_n \to t_0$. Then Γ is closed at $t_0 \in T$ if $x_n \in \Gamma(t_n)$ for each n and $x_n \to x_0$ together imply that $x_0 \in \Gamma(t_0)$.*

Theorem 4.1 *(2.2.2 from [23]) If f is a continuous real-valued function defined on the space $\mathbb{R}^m \times T$ and R is a continuous mapping of T into $\mathbb{R}^m$ such that $R(\epsilon) \neq \emptyset$ for each $\epsilon \in T$, then the (real-valued) function f^* defined by*

$$f^*(\epsilon) = \inf\{f(x, \epsilon) \mid x \in R(\epsilon)\} \tag{4.20}$$

is continuous on T.

If g is an affine function from $\mathbb{R}^n$ to $\mathbb{R}^m$, i.e., if $g(x) = Ax + b$, where A is an $m \times n$ constant matrix and $b \in \mathbb{R}^m$ is a constant vector, and if

$$R_M(g) = \{x \in M \mid g(x) \geq 0\},$$

where $M \subset \mathbb{R}^n$, the function g is said to be *non-degenerate* with respect to the set M if $R_M(g)$ has a non-empty interior and no component of g is identically zero on M. The continuity of the map defined by

$$S_M(g) = \{x \in R_M(g) \mid f(x) = \inf_z\{f(z) \mid z \in R_M(g)\}\} \tag{4.21}$$

is given by the following theorem. This result has been proven in [18].

Theorem 4.2 *(2.2.4 from [23]) If f is continuous, g is affine, and M is closed and convex, then S_M is closed at every non-degenerate g.*

We next apply these sensitivity results to our problem. Let us introduce a function $f_i(x_i, \epsilon)$ defined on $\mathbb{R} \times [0, \bar{\epsilon}]$ for every $i = 1, \ldots, n$ as follows

$$f_i(x_i, \epsilon) = \begin{cases} \bar{f}_i(x_i, \epsilon) & \text{if } \epsilon > 0, \\ f_i(x_i) & \text{if } \epsilon = 0, \end{cases} \tag{4.22}$$

where the functions $\bar{f}_i(x_i, \epsilon)$ are defined by (4.12) and the functions $s_{i\epsilon}(x)$ are defined by (4.13) or (4.16).

Finally, let

$$f(x, \epsilon) = F(x) + \sum_{i=1}^{n} f_i(x_i, \epsilon). \tag{4.23}$$

Consider the problem

$$\begin{aligned} \min \quad & f(x), \\ \text{s.t.} \quad & Ax \leq b. \end{aligned} \tag{4.24}$$

Proposition 4.1 *Let $f(x, \epsilon)$ be a function defined on $\mathbb{R}^n \times [0, \bar{\epsilon}]$ by (4.23) and let $\bar{\epsilon} > 0$ be given by (4.19). Then the optimal value function $f^*(\epsilon)$ defined by*

$$f^*(\epsilon) = \min\{f(x, \epsilon) \mid Ax \leq b\} \tag{4.25}$$

is continuous on $[0, \bar{\epsilon}]$. Furthermore, the mapping S, defined by

$$S(\epsilon) = \{x \mid Ax \leq b,\ f(x, \epsilon) = f^*(\epsilon)\} \tag{4.26}$$

is a closed mapping of $[0, \bar{\epsilon}]$ into $\mathbb{R}^n$.

Proof. We first show that $f_i(x_i, \epsilon)$ is continuous on $\mathbb{R} \times [0, \bar{\epsilon}]$ for every $i = 1, \ldots, n$. This is true for $\epsilon \neq 0$ because $\bar{f}_i(x_i, \epsilon)$ is a continuous function.

Let us show the continuity of $f_i(x_i, \epsilon)$ at $\epsilon = 0$. Consider a sequence $(x^k, \epsilon^k) \to (\bar{x}, 0)$. Suppose, first that $\bar{x}_i \neq d_{il}$ for any $l = 1, \ldots, M_i$. Then, for a sufficiently small value of ϵ^k, $x_i^k \notin [d_{il} - \epsilon_k, d_{il} + \epsilon_k]$ for any $l = 1, \ldots, M_i$. By definition of $f_i(x_i, \epsilon)$, $f_i(x_i^k, \epsilon^k) = f_i(x_i^k)$ and $f_i(\bar{x}_i) = f_i(\bar{x}_i, 0)$. Since $f_i(x_i)$ is continuous, $f_i(x_i^k) \to f_i(\bar{x}_i)$. Combining these, we obtain that $f_i(x_i^k, \epsilon^k) = f_i(x_i^k) \to f_i(\bar{x}_i) = f_i(\bar{x}_i, 0)$, or $f_i(x_i^k, \epsilon^k) \to f_i(\bar{x}_i, 0)$.

Suppose, next that $\bar{x}_i = d_{il}$ for some $l = 1, \ldots, M_i$. Then, for a sufficiently small value of ϵ^k, $x_i^k \in (d_{il-1} + \epsilon^k, d_{il+1} - \epsilon^k)$. We now subdivide (x^k, ϵ^k) into two subsequences. If $x_i^k \notin [d_{il} - \epsilon^k, d_{il} + \epsilon^k]$, then $f_i(x_i^k, \epsilon^k) = f_i(x_i^k) \to f_i(\bar{x}_i) = f_i(\bar{x}_i, 0)$ since $f_i(x_i)$ is continuous. If $x_i^k \in [d_{il} - \epsilon^k, d_{il} + \epsilon^k]$, then, in the case of a quadratic spline,

$$s_i(x_i^k) = \frac{\Delta p_{il}}{4\epsilon^k}(x_i^k - d_{il} + \epsilon^k)^2 \leq \frac{\Delta p_{il}}{4\epsilon^k}(2\epsilon^k)^2 \to 0, \tag{4.27}$$

when $\epsilon^k \to 0$. In the case of a cubic spline,

$$s_i(x_i^k) = \frac{\Delta p_{il}}{6(\epsilon^k)^2}(x_i^k - d_{il} + \epsilon^k)^3 \leq \frac{\Delta p_{il}}{6(\epsilon^k)^2}(2\epsilon^k)^3 \to 0, \tag{4.28}$$

when $\epsilon^k \to 0$. Therefore, $f_i(x_i^k, \epsilon^k) = f_i(x_i^k) + s_i(x_i^k) \to f_i(\bar{x}_i) = f_i(\bar{x}_i, 0)$.

This completes the proof of the fact that $f_i(x_i, \epsilon)$ is continuous for every $i = 1, \ldots, n$. Hence $f_i(x, \epsilon)$ is continuous as a sum of continuous functions.

Applying Theorem 4.1, we obtain (4.25).

We next reformulate the problem

$$\min\{f(x, \epsilon) \mid Ax \leq b\}$$

as

$$\min\{f(x, x_{n+1}) \mid Ax \leq b,\ x_{n+1} = \epsilon\}.$$

This allows us to apply Theorem 4.2, and proves (4.26). ■

By Proposition 4.1, approximating the piecewise linear functions $f_i(x_i)$ by smooth splines in the ϵ-neighborhood of the break-points for small values of ϵ implies small changes in the optimal solution and the optimal value of the problem.

Suppose (x^*, s^*, u^*, v^*) satisfy optimality conditions (2.6), i.e. $x^* \in S^*(0)$. From the optimality conditions (2.6),

$$\left.\begin{array}{ll}
(\nabla F(x^*))_i + p_{il} + (A^T u^*)_i = 0, & \text{for all } i \in N(x^*), \\
 & \text{with } x_i^* \in (d_{il}, d_{il+1}), \\
(\nabla F(x^*))_i + p_{il-1} + (A^T u^*)_i + v_i^* = 0, & \text{for all } i \in E(x^*), \\
 & \text{with } x_i^* = d_{il}, \\
Ax^* + s^* = b,\ s^* \geq 0 & \\
0 \leq v_i^* \leq p_{il} - p_{il-1}, & \text{for all } i \in E(x^*), \\
 & \text{with } {}^*x_i = d_{il}, \\
u^* \geq 0, & \\
u_i^* s_i^* = 0,\ i = 1, \ldots, m. &
\end{array}\right\} \tag{4.29}$$

Suppose next that $(x^* + \Delta x, s^* + \Delta s, u^* + \Delta u)$ are optimal for the problem (4.24) for a given $\epsilon > 0$, i.e. $x^* \in S^*(\epsilon)$. Let us denote $f_\epsilon(x) = f(x, \epsilon)$. The optimality conditions for this problem imply

$$\left.\begin{array}{l}
\nabla f_\epsilon(x^* + \Delta x) + A^T(u^* + \Delta u) = 0, \\
A(x^* + \Delta x) + (s^* + \Delta s) = b,\ s^* + \Delta s \geq 0, \\
(u^* + \Delta u) \geq 0, \\
(u_i^* + \Delta u_i)(s_i^* + \Delta s_i) = 0,\ i = 1, \ldots, m.
\end{array}\right\} \tag{4.30}$$

Approximating the gradient of $f_\epsilon(x)$ by it's Taylor series, we obtain

$$\left.\begin{array}{l}
\nabla f_\epsilon(x^*) + \nabla f_\epsilon^2(x^*)\Delta x + A^T(u^* + \Delta u) = 0, \\
A(x^* + \Delta x) + (s^* + \Delta s) = b,\ s^* + \Delta s \geq 0, \\
(u^* + \Delta u) \geq 0, \\
(u_i^* + \Delta u_i)(s_i^* + \Delta s_i) = 0,\ i = 1, \ldots, m,
\end{array}\right\} \tag{4.31}$$

or

$$\left.\begin{array}{ll}
(\nabla F(x^*))_i + p_{il-1} + (\nabla s_\epsilon(x^*))_i + (\nabla F^2(x^*)\Delta x)_i + (\nabla s_\epsilon^2(x^*)\Delta x)_i + & \text{for all } i \in E(x^*), \\
(A^T(u^* + \Delta u))_i = 0, & \text{with } x_i^* = d_{il}, \\
(\nabla F(x^*))_i + p_{il-1} + (\nabla F^2(x^*)\Delta x)_i + (A^T(u^* + \Delta u))_i = 0, & \text{for all } i \in N(x^*), \\
 & \text{with } x_i^* \in [d_{il-1}, d_{il}], \\
A(x^* + \Delta x) + (s^* + \Delta s) = b,\ s^* + \Delta s \geq 0, & \\
(u^* + \Delta u) \geq 0, & \\
(u_i^* + \Delta u_i)(s_i^* + \Delta s_i) = 0,\ i = 1, \ldots, m. &
\end{array}\right\} \quad (4.32)$$

Subtracting system (4.29) from (4.32) we get

$$\left.\begin{array}{ll}
(\nabla s_\epsilon(x^*))_i + (\nabla F^2(x^*)\Delta x)_i + (\nabla s_\epsilon^2(x^*)\Delta x)_i - v_i + & \text{for all } i \in E(x^*), \\
(A^T(\Delta u))_i = 0, & \text{with } x_i^* = d_{il}, \\
(\nabla F^2(x^*)\Delta x)_i + (A^T(\Delta u))_i = 0, & \text{for all } i \in N(x^*), \\
 & \text{with } x_i^* \in [d_{il-1}, d_{il}], \\
A\Delta x + \Delta s = 0, & \\
u_i^* * \Delta s_i + s_i^* * \Delta u_i + \Delta s_i * \Delta u_i = 0,\ i = 1, \ldots, m. &
\end{array}\right\} \quad (4.33)$$

If we disregard the last term of the last equation $\Delta s_i * \Delta u_i$, we can obtain $(\Delta x, \Delta s, \Delta u)$ by solving a linear system

$$\begin{bmatrix} \nabla F^2(x^*) + H & A^T & 0 \\ A & 0 & I \\ 0 & S & U \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta u \\ \Delta s \end{bmatrix} = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}, \quad (4.34)$$

where

$$r_i = \begin{cases} v_i - \nabla s_\epsilon(x^*)_i & \text{if } \ x_i^* = d_{il}, \\ 0, & \text{otherwise}, \end{cases} \quad (4.35)$$

and H is a diagonal matrix

$$H_{ii} = \begin{cases} \nabla s_\epsilon^2(x^*)_i & \text{if } \ x_i^* = d_{il}, \\ 0, & \text{otherwise}. \end{cases} \quad (4.36)$$

This proves the following:

Theorem 4.3 *Let $\epsilon > 0$ and $x(\epsilon)$ be an optimal solution for problem(4.24) and let $s(\epsilon)$ and $u(\epsilon)$ be the slack variables and dual variables associated with this optimal solution. Then there exists an optimal solution x^* for problem(2.1) such that the first-order approximation $(x(\epsilon), s(\epsilon), u(\epsilon))$ in a neighborhood of $\epsilon = 0$ is given by*

$$(x(\epsilon), s(\epsilon), u(\epsilon)) = (x^*, s^*, u^*) + (\Delta x, \Delta s, \Delta u) + o(\epsilon),$$

where s^* and u^* are the slack variables and dual variables associated with x^*, and $(\Delta x, \Delta s, \Delta u)$ can be found from (4.34).

■

Note that the LHS of the system (4.34) is similar to (4.6).

If we assume strict complementarity at x^*, the last line of (4.34) implies that the active set will not change for small values of ϵ. Let us denote by $\bar{A}$ the matrix of the constraints active at x^*. Then the system (4.34) can be rewritten as follows.

$$\begin{bmatrix} \nabla F^2(x^*) + H & \bar{A}^T \\ \bar{A} & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}, \tag{4.37}$$

We further note that the norm of H is equal to a constant multiple of $\max_{i,l} \frac{\Delta p_{il}}{\epsilon}$ for both quadratic and cubic splines. For small values of ϵ it can be used as an estimate of the norm of the LHS matrix of (4.37). Also, $0 \leq r_i \leq \max_l \Delta p_{il}$. Hence, we expect that the norm of Δx will be order ϵ. This is consistent with our computational experiments.

4.5 Smooth Formulations via Lifting

Because of the special structure of the nondifferentiable part of the objective function, problem (2.1) can be converted to a smooth one by introducing new variables and constraints into the problem. For example, for each $i = 1, \ldots, n$, we can introduce a new set of variables x_{il}^+ where $l = 0, \ldots, M_i^+$ and x_{il}^- where $l = 0, \ldots, M_i^-$ .Then problem (2.1) can be rewritten as (1.6).

The problem (1.6) is a linearly constrained, convex and twice differentiable problem and standard techniques can be used to solve it. However the higher dimensional problem is computationally expensive to solve.

We can design an interior-point algorithm for this problem in a way analogous to our derivation in Subsection 3.1. First, we express the primal-dual central path. The central path is continuously differentiable, however, it is expressed in a very high dimensional space compared to our formulations in Section 3. To compare the underlying interior-point algorithms, we can eliminate the "new variables" (those not present in the formulations of Section 3) from the nonlinear system defining the central path. Let $v \in \mathbb{R}^n$ denote the dual variables corresponding to the linear equations expressing x in terms of $\hat{x}, x^+$, and x^-.

After eliminating all of these new variables except v, the nonlinear system of equations and inequalities are equivalently written as

$$\begin{aligned} Gx + c + A^T u + v(x) &= 0, \quad u > 0; \\ Ax + s &= b, \quad s > 0; \\ Su &= \mu e. \end{aligned}$$

In the above $v(x) : \mathbb{R}^n \to \mathbb{R}^n$ is continuously differentiable at all interior points and is completely separable, that is, $[v(x)]_i$ only depends on x_i. Therefore, if we

derive the search directions based on this latest system, we end up with the normal equations determining Δx whose coefficient matrix is:

$$G + \text{Diag}\,[v'(x)] + A^T(S^{-1}U)A.$$

Compared to the search directions from Section 3, the search direction derived here has only a diagonal perturbation to the left-hand-side matrix and this perturbation $\text{Diag}\,[v'(x)]$ is independent of A, b, c, G and only depends on the nondifferentiable part of the data.

Another approach to comparing different interior-point algorithms in our setting would be to derive the search directions for each formulation in their own space and then consider the Δx components of each of these search directions. I.e., compare the projections of the search directions in the x-space. This way of comparing the search directions could lead to different conclusions than the above. As it will become clear, our way of comparing these different formulations exposes the similarities in an extremely easy and elegant way. One drawback of the compact central path system that we derived is, the compact system is "more nonlinear" than the high dimensional, original formulation. The latter is the usual approach to quadratic programming and yields a bilinear system of equations and strict linear inequalities. In our compact system above, in addition to the usual bilinear equations (such as $Su = \mu e$), we also have the nonlinear system involving $v'(x)$.

4.6 Local Lifting

We can introduce a pair of new variables $x_i^+,\ x_i^-$ for each $i \in E(x)$ with $x_i = d_{il}$. This converts a problem into a differentiable one in the neighborhood of the current

iterate. The problem becomes

$$\begin{array}{ll} \min & F(x) + \sum_{i \in N(x)} f_i(x_i) + \sum_{i \in E(x)} (f_{il}^+(x_i^+) + f_{il}^-(x_i^-)), \\ \text{s.t.} & Ax \leq b, \\ & x_i = d_{il} + x_i^+ - x_i^-, \ i \in E(x), \\ & x_i^+, \ x_i^- \geq 0, \ i \in E(x). \end{array} \tag{4.38}$$

A point $x \in \mathbb{R}^n$ is a point on a central path corresponding to (4.38), if and only if

$$\left.\begin{array}{ll} Ax + s = b, \ s \geq 0 & \\ (\nabla F(x))_i + \frac{df_{il}}{dx_i}(x_i) + (A^T u)_i = 0, & \text{for all } i \in N(x), \\ & \text{with } x_i \in (d_{il}, d_{il+1}), \\ (\nabla F(x))_i + \frac{df_{il-1}}{dx_i}(d_{il}) + (A^T u)_i + v_i = 0, & \text{for all } i \in E(x), \\ & \text{with } x_i = d_{il}, \\ (\nabla F(x))_i + \frac{df_{il}}{dx_i}(d_{il}) + (A^T u)_i - w_i = 0, & \text{for all } i \in E(x), \\ & \text{with } x_i = d_{il}, \\ u \geq 0, & \\ x_i = d_{il} + x_i^+ - x_i^-, \ i \in E(x), & \\ x_i^+, \ x_i^- \geq 0, \ i \in E(x), & \\ v_i, \ w_i \geq 0, \ i \in E(x), & \\ u_i s_i = \mu, \ i = 1, \ldots, m, & \\ v_i x_i^- = \mu, \ i \in E(x), & \\ w_i x_i^+ = \mu, \ i \in E(x). & \end{array}\right\} \tag{4.39}$$

Note that $v_i + w_i = \Delta p_{il}$. For the quadratic function, the above system becomes

$$\left.\begin{array}{ll}
Ax + s = b,\ s \geq 0 & \\
(Gx)_i + c_i + p_{il} + (A^T u)_i = 0, & \text{for all } i \in N(x), \\
 & \text{with } x_i \in (d_{il}, d_{il+1}), \\
(Gx)_i + c_i + p_{il-1} + (A^T u)_i + v_i = 0, & \text{for all } i \in E(x), \\
 & \text{with } x_i = d_{il}, \\
u \geq 0, & \\
u_i s_i = \mu, & i = 1, \ldots, m, \\
x_i^+, \geq 0,\ x^- \geq 0, & i \in E(x), \\
w_i \geq 0,\ v_i \geq 0, & i \in E(x), \\
v_i + w_i = \Delta p_{il}, & i \in E(x), \\
x_i = d_{il} + x_i^+ - x_i^-, & i \in E(x), \\
v_i x_i^- = \mu, & i \in E(x), \\
w_i x_i^+ = \mu, & i \in E(x).
\end{array}\right\} \tag{4.40}$$

The last four equalities form a system

$$\left.\begin{array}{l}
v_i + w_i = \Delta p_{il}, \\
x_i = d_{il} + x_i^+ - x_i^-, \\
v_i x_i^- = \mu, \\
w_i x_i^+ = \mu.
\end{array}\right\} \tag{4.41}$$

This allows us to express the dual variable v_i as a function of x_i

$$v_i(x_i) = \frac{2\mu\Delta p_{il}}{2\mu - \Delta p_{il}(x_i - d_{il}) + (4\mu^2 + \Delta p_{il}^2 (x_i - d_{il})^2)^{\frac{1}{2}}}. \tag{4.42}$$

Note that $v_i(d_{il}) = \frac{\Delta p_{il}}{2} > 0$. In a neighborhood of d_{il} the variables w_i, x_i^+ and

x_i^- are positive and solving a system (4.40) is equivalent to solving

$$\left.\begin{array}{ll} Ax + s = b,\ s \geq 0 & \\ (Gx)_i + c_i + p_{il} + (A^T u)_i = 0, & \text{for all } i \in N(x), \\ & \text{with } x_i \in (d_{il}, d_{il+1}), \\ (Gx)_i + c_i + p_{il-1} + (A^T u)_i + v_i(x_i) = 0, & \text{for all } i \in E(x), \\ & \text{with } x_i = d_{il}, \\ u \geq 0, & \\ u_i s_i = \mu, & i = 1, \ldots, m. \end{array}\right\} \tag{4.43}$$

This approach appears to be similar to the "spline approximation" approach. We are just modeling the jumps in the gradient of $f_i(x_i)$ by a function $s_i(x_i)$, such that

$$s_i'(x_i) = v_i(x_i).$$

Proposition 4.2 *Suppose an interior point method is applied to the problem (4.38) and a search direction $(\Delta x, \Delta u, \Delta s)$ is obtained at a point (x, u, s). Then the same direction can also be obtained by applying the interior point method to the problem (4.5), with $\bar{f}_i(x_i) = f_i(x_i) + s_i(x_i)$, where $s_i'(x_i) = v_i(x_i)$ is given by (4.42).*

Therefore, the search direction computed in this Local Lifting approach can also be treated in the class of search directions Δx obtained for m solving the system

$$[G + D + A^T(S^{-1}U)A]\Delta x \ = -(Gx + c + d) - A^T S^{-1}[Ur_b + \sigma\mu e]. \tag{4.44}$$

where D and d are the diagonal matrix and a vector determined by a particular approach (e.g., smoothing via quadratic spline, smoothing via cubic spline, global lifting, local lifting).

Note that the above unification of these various approaches go even deeper. In subsection 4.4, the sensitivity analysis leads to the linear system of equations (4.34) which is also in the above form.

The main reason for this is the fact that we derived our algorithm from the necessary and sufficient conditions for optimality and these conditions change slightly going form one of our formulations to another.

4.7 Probability Analysis for Number of Break Points

Recall that $f : \mathbb{R}^n \to \mathbb{R}^n$ is a convex function which is the sum of a strictly convex quadratic function and n, convex, separable, piecewise linear functions. So, f is differentiable everywhere except at the break points of the piecewise linear components.

The problem we have been considering is

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in P, \end{aligned}$$

where $P \subseteq \mathbb{R}^n$ is a polyhedron.

For every $z \in \mathbb{R}$, define the level set of f:

$$C(z) := \{x \in \mathbb{R}^n : f(x) \leq z\},$$

which is convex (since f is) and closed (since f is continuous). Suppose that our optimization problem has an optimal value z^* and it is attained. Then we ask the

question"How likely is it that there exist

$$\bar{x} \in C(z^*) \cap P \text{ and } i \in \{1, 2, \ldots, n\} \text{ such that}$$

$\bar{x}_i$ is a break point of f_i?"

Proposition 4.3 *Let f, $C(z)$, P, and z^* be as defined above. Then, $C(z)$ is a compact, convex set for every $z \in \mathbb{R}^n$. Moreover, if $P \neq \emptyset$, then z^* exists and is attained by a unique $x^* \in P$.*

Proof: We already established that $C(z)$ is closed and convex for every $z \in \mathbb{R}^n$. Since f is the sum of a strictly convex quadratic function and n, convex, piecewise linear functions, f is coercive. Thus, $C(z)$ is bounded for every $z \in \mathbb{R}^n$. Therefore, $C(z)$ is compact for every $z \in \mathbb{R}^n$. We deduce that if $P \neq \emptyset$, then z^* is finite and is attained. f is strictly convex (since it is the sum of a strictly convex function and some other convex functions); hence, there must be a unique minimizer of f on the compact, convex set $C(z^*) \cap P$. ■

In our analysis, restricted to the domain of our applications, we may assume $0 \in P$. Recall that $\hat{x}$ denotes the current investment holdings and we would expect it to be feasible. (By a simple translation of the coordinate system, we can place $\hat{x}$ at the origin.) Now, for each j, $x_j > 0$ represents buying and $x_j < 0$ represents selling. Since neither of these two activities is free, we conclude that each piecewise linear function has a break point at zero. Therefore, the objective function f is nondifferentiable on the hyperplanes,

$$\{x \in \mathbb{R}^n : x_j = 0\}$$

for every j.

From a practical viewpoint, we immediately have an answer to our question. Since the investor cannot be expected to trade every single stock/commodity in every planning horizon, break points at optimality are unavoidable!

From the theoretical viewpoint the answers depend on the probabilistic model used and calculating the probabilities exactly would be complicated.

In order to get a rough estimate of the number of the coordinates of the optimal solution that are at breakpoints, we looked at a simpler problem of unconstrained minimization of $f(x)$. In addition, we assumed the the matrix G is diagonal, functions $f_i(x_i)$ are the same for each coordinate, the breakpoints d_{il} and the gradients p_{il} are equally spaced. We denote by $\Delta d = d_{il+1} - d_{il}$ and by $\Delta p = p_{il+1} - p_{il}$. From the optimality conditions (2.6), $\bar{x}$ minimizes $f(x)$ if and only if $0 \in \partial f(x)$ or

$$0 \in G_i x_i + c_i + \partial f_i(x_i),\ i = 1, \ldots, n.$$

We can think of a subdifferential as a mapping from $\mathbb{R}^n \to \mathbb{R}^n$. If none of the coordinates of x are at breakpoints, this point is mapped to a single point. If a coordinate $x_i = d_{il}$ is at a breakpoint, then the i-th component of the subdifferential is an interval and the probability of having $x_i = d_{il}$ is equal to the probability of zero being in this interval.

If the coordinate $x_i = d_{il}$ is at a breakpoint,

$$0 \in [G_i d_{il} + c_i + p_{il-1}, G_i d_{il} + c_i + p_{il}].$$

Note that the length of this interval is equal to Δp.

If the coordinate $x_i \in (d_{il}, d_{il+1})$ is not at a breakpoint,

$$0 = G_i x_i + c_i + p_{il}.$$

The interval (d_{il}, d_{il+1}) is mapped to an interval

$$[G_i d_{il} + c_i + p_{il}, G_i d_{il+1} + c_i + p_{il}]$$

of length $G_i \Delta d$.

This suggest a hypothesis that ratio of the probability of the i-th coordinate being at a breakpoint to the probability of the opposite event is

$$\Delta p : G_i \Delta d.$$

We ran some tests for the constrained minimization of such functions. In some examples this estimate was pretty close, in other cases the number of the breakpoints was less than predicted, see Table 5.6 in the next section.

Chapter 5

Computational Experiments

Both interior-point and active set algorithms are implemented for the case when the utility function is quadratic and transaction costs are piece-wise linear. To be more precise, we considered the problem

$$\begin{array}{ll} \min & \frac{1}{2}x'Gx + c'x + \sum_{i=1}^{n} f_i(x_i) \\ s.t. & x_1 + x_2 + ... + x_n = 1, \\ & Ax \leq b, \end{array} \tag{5.1}$$

where

$$f_i(x_i) = \begin{cases} p_{i0}x_i + h_{i0}, & \text{if} \quad x_i \leq d_{i1}, \\ p_{il}x_i + h_{il}, & \text{if} \quad x_i \in [d_{il}, d_{il+1}], \quad l = 1, .., M_i, \\ p_{iM_i}x_i + h_{iM_i}, & \text{if} \quad x_i \geq d_{iM_I}. \end{cases} \tag{5.2}$$

The IPM algorithm is implemented in MATLAB and the active set method is implemented in C. In Section 5.1 we test the performance of the active set algorithm. In Section 5.2 we show how the parameters of the problem affect the performance of the IPM algorithm. In Section 5.3 we look at the connection between these parameters and the number of the coordinates of the optimal solution x_i that have

values at points of nondifferentiability. A crossover from the IPM to an active set algorithm is tested in Section 5.4. For all of the above experiments medium-scale dense data is used.

We also tested the IPM algorithm on large-scale sparse data. These results are reported in Section 5.6 and some implementation details are discussed in Section 5.5.

The data is randomly generated in the following way. Vector c corresponds to the vector of the expected returns, randomly generated in the range (1, 1.3). The target vector $\hat{x}$ is set to zero. The number of the points of nondifferentiability $M_i = M$ is the same for each coordinate. The transaction costs are chosen as follows.

$$f_i(x_i) = \begin{cases} p_{min}x_i + h_{i0}, & \text{if} \quad x_i \leq d_{min}, \\ (p_{min} + \frac{(p_{max}-p_{min})l}{M-1})x_i + h_{il}, & \text{if} \quad d_{min} + \frac{(d_{max}-d_{min})(l-1)}{M-1} \leq x_i \\ & \qquad \leq d_{min} + \frac{(d_{max}-d_{min})l}{M-1}, \\ p_{max}x_i + h_{iM}, & \text{if} \quad x_i \geq d_{max}. \end{cases} \tag{5.3}$$

We'll say that the transaction costs varied from p_{min} to p_{max} in this case. The matrix $G = \alpha\bar{G}$. Roughly speaking, α is proportional to the inverse of the risk aversion parameter t, i.e., higher values of α correspond to lower values of t and vice versa. We discuss the effect of changing this constant on the problem in Section 5.2.

1. **Dense Data.** In order to guarantee that the matrix G is positive semi-definite, we first construct an $n \times n$ matrix C with random entries in the range (-0.5, 0.5) and then form $\bar{G} = C^TC$. The matrix of the inequality constraints, A, and the vector of the right hand side, b, are also generated randomly. In the first series of experiments we generate A and b with random entries in the range (-0.5, 0.5). We refer to this kind of data as Type 1. In

the second series of experiments we generate A with random integer entries from the set $\{0, 1, 2, 3\}$. We refer to this kind of data as Type 2. Each dense problem has one equality constraint $x_1 + x_2 + \cdots + x_n = 1$. The transaction costs varied form -0.5 to 0.5.

2. **Sparse Data.** We test 2 types of sparse data. We use the *sprandsym* command in MATLAB to generate the matrices $\bar{G}$ with a given sparsity. Another type of data arising in large-scale applications is block-diagonal matrices or matrices with overlapping blocks on the diagonal. We use *sprandsym* command in MATLAB to generate each of the blocks. The transaction costs varied form -0.05 to 0.05. The matrix of the inequality constraints, A, is also sparse. In all the experiments, we ensure that A has no zero column or zero row. If the randomly generated matrix A has one or more zero columns, then we add a random number to one element in each of these columns, but in a different row. Then we check the resulting matrix for zero rows and eliminate them in a similar way.

In order to compare the performance of our implementation with that of commercial package, we convert our problem into a differentiable one (1.6) by introducing nM new variables $x^{1+}, x^{2+}, \ldots, x^{M+}$, $x^{1-}, x^{2-}, \ldots, x^{M-}$. This problem is then solved using MOSEK 3, using an interior point method and in CPLEX 9.0 using the active set method with default accuracy.

We run all the experiments on a SUNW, UltraSparc-IIIi, (1002 MHz, 2048 Megabytes of RAM). All execution times are given in CPU seconds. We repeat each experiment 10 times for the smaller dense problems and 5 times for the large sparse ones. The average execution times are reported in each table. The requested accuracy for our MATLAB code is $\epsilon/100$, where ϵ is the parameter of the spline

approximation. In Section 5.6 we request the same accuracy from MOSEK. In Sections 5.1 and 5.4 , where the active set and crossover method are tested, the relative gap termination tolerance for MOSEK was increased to 10e-14.

5.1 Active Set Algorithm

The active set method is implemented in C for the problem in a standard equality form. At each iteration of the active set method, the variables are subdivided into basic and non-basic. A coordinate can be non-basic only if it is equal to one of the breakpoints (if upper or lower bounds are present, they are treated as breakpoints too). A Newton's search direction is taken in the basic subspace. This requires solving a symmetric linear system at each iteration. The step size is taken so that all the variables stay in the corresponding intervals between the breakpoints. At the end of each iteration either one variable is added to the basis or it is dropped from the basis. We store the inverse of the coefficient matrix of the system. The next linear system differs form the previous one by one row and column. We use this fact to update the inverse. The FORTRAN routines for this updates were kindly provided by Professor M.J.Best. These routines were converted to C by an automatic translator. To further improve efficiency we used CBLAS routines to perform basic matrix and vector operations.

These experiments were performed on the dense data Type 2. Each problem had equality constraints and upper and lower bounds on all variables. In all the experiments CPLEX and our Algorithm 2 were started with the same initial basis. Note that for CPLEX we give both total time spent on creating the model and solution and pure execution time. For the remaining methods these timings were virtually the same so only total time is given. In Tables 5.1 and 5.2 the transaction

	CPLEX 9.0	MOSEK 3	Algorithm 2
n=600			
m=1	26/23 (969)	6 (15)	15 (806)
m=30	26/23 (864)	25 (14)	18 (811)
n=1000			
m=1	123/107 (1398)	15 (15)	88 (975)
m=50	111/ 94 (1078)	110 (15)	106 (866)
n=1400			
m=1	355/316 (1955)	36 (16)	305 (1288)
m=70	310/271 (1407)	351 (18)	362 (1148)

Table 5.1: CPU times for Algorithm 2, CPLEX and MOSEK, M=3.

cost function has 3 breakpoints, the costs are 0.5% for the first interval and 1.5% for the second one. For Table 5.1 the initial point for the problem is chosen so that it is feasible and $x_i \approx \frac{1}{n}$.

In practice, a portfolio is rebalanced regularly to reflect the changes in the market data. Unless the data changes significantly, the current portfolio (target portfolio), which was optimal before the change, is a very natural choice for the initial point for an active set method. In our computations we assume that only coefficients of the objective function change, while the constraints remain unchanged. Under this condition, the target portfolio is feasible for our problem. However this point is always degenerate, since all it's coordinates are at breakpoints and all the equality constraints are always active. We noticed that for the case of one equality constraint Algorithm 2 performs well. Otherwise we solve the LP described in Section 3.5. We report the time for this LP in a separate column. This time is also included in the total time for the Algorithm 2. Note that LP accounts for

	CPLEX 9.0	Algorithm 2	LP	At Breakpoints
n=600				
m=1	6/3 (325)	1 (327)	-	365
m=30	7/4 (472)	4 (231)	3	354
n=1000				
m=1	23/9 (374)	4 (370)	-	644
m=50	29 /15 (614)	19 (187)	15	666
n=1400				
m=1	57/22 (467)	9 (468)	-	941
m=70	70/34 (885)	71 (246)	60	940

Table 5.2: CPU times for Algorithm 2 and CPLEX, warm start.

CPLEX 9.0	Algorithm 2	At Breakpoints
23/9 (374)	4 (370)	644
20/6 (228)	1 (229)	781
18/4 (152)	0.4 (153)	857

Table 5.3: CPU times for Algorithm 2 and CPLEX, warm start, n=1000, m=1, M=3.

approximately 80% of the total time in these cases. Finding a more efficient way of dealing with degeneracy would give a big improvement. Table 5.2 shows the "warm start" timings for CPLEX and Algorithm 2. In the " warm start" situation described above only a small percentage of the assets are traded, the rest stay at the original level (breakpoint). We noticed that increasing the transaction cost forces more assets to stay at the breakpoints and Algorithm 2 performs best in these cases, see Table 5.3.

5.2 Number of break points M; Spline Neighborhood ϵ

1. We tested the MATLAB IPM program in Table 5.4 and varied the number of break points M from 3 to 101 and the size of the spline intervals ϵ from 0.001 to 0.00001. The transaction costs varied form -0.5 to 0.5. Figure 5.1 illustrates the timings for the cubic spline case.

   We can see that increasing ϵ consistently decreases the number of iterations and cpu time. (Though our theoretical sensitivity results show that the accuracy to the true optimum decreases, see Section 4.4). Also, increasing the number of intervals decreases the number of iterations and CPU time. In both cases, the problem becomes more like a smooth problem.

   We noticed that the difference between the real optimlsolution and the solution to the spline approximation is always less than ϵ.

2. We also ran our IPM code on the same set of problems, changing only the value of the constant α in the definition of the matrix G. All the remaining parameter are fixed: cubic spline, M=51, $\epsilon = 0.0001$, n=1000, m=500. We noticed that decreasing $alpha$ increases the timings for the problem with transaction costs, see Table 5.5. We also report the expected return for the optimal solutions of the problems with transaction costs. Note that smaller values of α correspond to larger values of the risk aversion parameter, for example $\alpha = 0.05$ gives an extremely risky portfolio with expected return of 321%. In all the further experiments we use the value of α that gives realistic expected returns.

M \ ϵ	0.001	0.0005	0.0001	0.00005	0.00001
	Quadratic Spline				
3	44 (20)	55 (23)	109 (38)	148 (46)	407(98)
25	36 (17)	38 (17)	52 (21)	61 (23)	107 (31)
51	36 (17)	37 (17)	43 (19)	52 (20)	87 (28)
75	36 (16)	37 (17)	41 (18)	46 (19)	77 (26)
101	35 (16)	38 (17)	43 (19)	45 (19)	71 (25)
	Cubic Spline				
3	43 (20)	53 (24)	97 (39)	133 (48)	348 (104)
25	35 (16)	37 (17)	49 (21)	59 (24)	98 (33)
51	34 (16)	36 (17)	44 (20)	49 (21)	84 (30)
75	33 (16)	35 (16)	42 (19)	47 (20)	70 (26)
101	33 (15)	35 (16)	42 (19)	45 (20)	71 (26)

Table 5.4: CPU time (iterations) for MATLAB IPM ; $n = 1000, m = 500$.

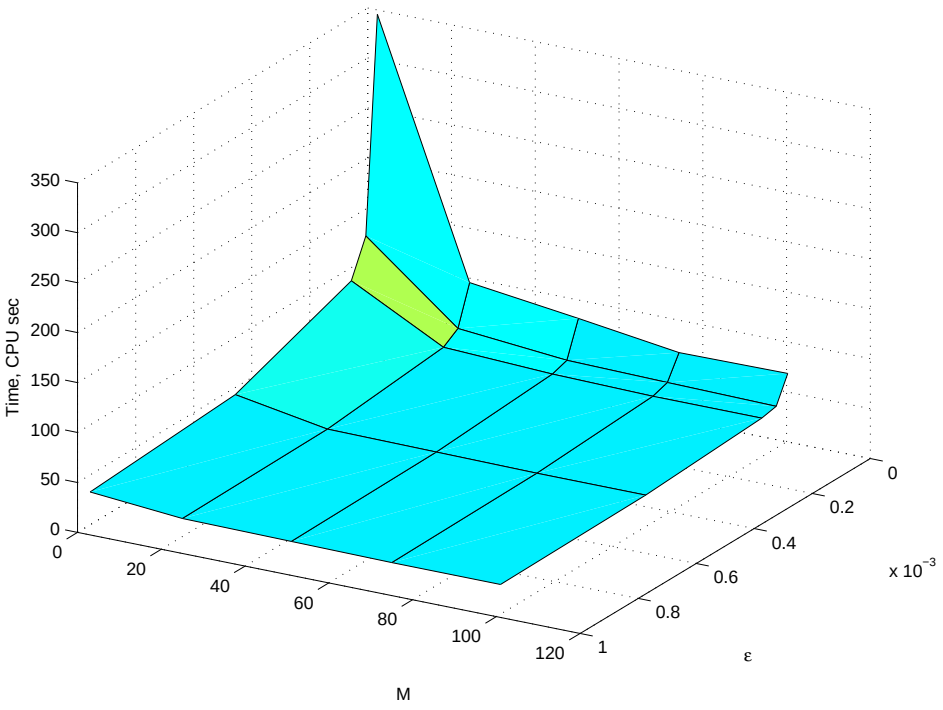

Figure 5.1: CPU time for MATLAB IPM ; $n = 1000, m = 500$, cubic spline.

	α=1	α=0.5	α=0.1	α=0.05
Problem with Trans. Costs	43 (20)	51 (22)	129 (46)	216 (74)
Expected return	1.35	1.56	2.46	3.21

Table 5.5: CPU time (iterations) for MATLAB IPM ; $n = 1000, m = 500$, $M = 101$, $\epsilon = 0.0001$.

5.3 Expected Number of Breakpoints

Following the probability analysis in Section 4.7, we would like to see how the parameters of the problem influence the number of the coordinates of the optimal solution coinciding with a breakpoint. For this set of experiments the Hessian matrix G is always diagonal. We first take $G = \alpha I$, i.e., G is a multiple of the identity matrix. The matrix of the linear system A has random entries in the range (-0.5, 0.5), the vector of the right hand sides b has random entries in the range (0, 1). Note that zero is always feasible for this problems. The rest of the data is generated as described above. The results are presented in Table 5.6.

In the Table 5.7 matrix G has $4, 3, 2$ or 1 on the diagonal, 25% of each. This subdivides the set of all coordinates into 4 groups. The rest of the data is as above. This table shows the number of coordinates of the optimal solution at a breakpoint in each of these subgroups.

	$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
$\Delta p = \Delta d$				
Experiment	167 (42%)	110 (28%)	68 (17%)	48 (12%)
Predicted	50%	33%	20%	11%
$\Delta p = 2\Delta d$				
Experiment	232 (58%)	179 (45%)	122 (31%)	78 (20%)
Predicted	66%	50%	33%	20%
$2\Delta p = \Delta d$				
Experiment	109 (27%)	72 (18%)	33 (8%)	21 (5%)
Predicted	33%	20%	11%	6%

Table 5.6: Number (percentage) of coordinates of the optimal solution at a breakpoint, n=400, m=800.

	G_{ii}=4	G_{ii}=3	G_{ii}=2	G_{ii}=1
Experiment	18(18%)	23(23%)	30(30%)	39(39%)
Predicted	20%	25%	33%	50%

Table 5.7: Number(percentage) of coordinates of the optimal solution at a breakpoint in each subgroup, n=400, m=800, $\Delta p = \Delta d$.

5.4 Crossover

If a more accurate solution is needed, we can do a crossover to an active set algorithm.

Two versions of crossover method between the IPM and active set method was tested.

In the first type of crossover we used the last iterate of the interior point method as an initial point for the active set method. However, because of the nature of the active set method, starting it with an interior point makes all the slack variables basic. The number of iterations needed for the active set method to finish the problem is at least the number of the constraints active at the optimum. Since our active set algorithm takes a Newton's step at each iteration , this method will be time consuming. It could perform well if only few constraints are active at the optimum and few coordinates are at breakpoints.

Another approach is to take a purification step first. We also use the last iterate of the interior point method as an initial point and we perform several iterations of the gradient projection method, see, for example, [42]. We stop if the optimal solution is found or if a constraint should be dropped. In the latter case the last iterate of the purification step is used to start an active set algorithm.

The purification step was implemented in MATLAB. At each iteration, we keep track of the set of active constraints and the projection matrix corresponding to these constraints. We find the orthogonal projection of the negative gradient of the objective function at the current iterate onto the subspace parallel to the affine space of the currently active constraints. We find the maximal feasible step size in this direction and also perform a line search to minimize the true objective function, along this direction. We either add one more constraint to the active set and update

the projection matrix, or stop, depending on the outcome of the line search. This method has the guarantee that the true objective function value of the final solution from the purification is at least as good as that of the final IPM solution. This method performed best in most cases. These results are summarized in Tables 5.8 and 5.9. From the Table 5.8, doing the purification step before the crossover is always faster. We only present the faster option it the remaining table, the number of iterations is given in brackets.

When the number of the breakpoints is large, our program is faster than MOSEK. We found that terminating the approximated problem when the relative gap is equal to ϵ gives slightly better timings. Also note that our IPM was implemented in MATLAB and is slower than MOSEK. We ran MOSEK and our code on the same differentiable problems, and MOSEK was approximately 2.5 times faster than our MATLAB code.

5.5 Linear System Solvers

We compared the MATLAB CPU times for the following three ways of solving the linear system to compute the search directions for the interior-point algorithms:

1. **Chol.** Form a sparse matrix $A^T(S^{-1}U)A$, and perform Choleski decomposition on a matrix
$$\left[G + H + A^T(S^{-1}U)A\right] \tag{5.4}$$
converted into dense format.

2. **Aug.** Directly solve the *augmented system* whose coefficient matrix is
$$\begin{bmatrix} G+H & A^T \\ A & -U^{-1}S \end{bmatrix}$$

				MOSEK	102 (25)
	MATLAB IPM	Purification Step(MATLAB)	ASet after Pur.	ASet after IPM	Crossover total
tol=10e-3					
With Pur.Step	24 (10)	18 (250)	85 (65)		127
No Pur.Step	24 (10)	-		430 (333)	
tol=10e-4					
With Pur.Step	32 (14)	18 (250)	50 (32)		100
No Pur.Step	32 (14)	-		390 (281)	
tol=10e-5					
With Pur.Step	35 (16)	18 (246)	48 (30)		101
No Pur.Step	35 (16)	-		389 (278)	

Table 5.8: CPU time (iterations) for Crossover and MOSEK, n=1000, m=500, M=101, ϵ=0.0001, Data Type 1.

			MOSEK	217 (31)
MATLAB Term. Tol.	MATLAB IPM	Purification Step(MATLAB)	ASet after Pur.	Crossover total
10e-3	25 (11)	18 (247)	76 (56)	119
10e-4	34 (15)	18 (248)	52 (33)	104
10e-5	36 (16)	18 (248)	57 (37)	111

Table 5.9: CPU time (iterations) for Crossover with purifiction step and MOSEK, n=1000, m=500, M=101, ϵ=0.0001, Data Type 2.

using the 'backslash'.

3. **Bcksl.** Solve the sparse system whose coefficient matrix is $\left[G + H + A^T(S^{-1}U)A\right]$ using the 'backslash'.

4. **Block LU** Solve the *augmented system* using a block LU approach. We first compute the LU factorization of the upper left block $G + H = L_{11}U_{11}$, then solve multiple rhs triangular systems $L_{11}U_{12} = A'$ and $L_{21}U_{11} = A$ for U_{12} and L_{21} respectively. Finally we form a matrix $S = -U^{-1}S - L_{21}U_{12}$ (a Schur compliment of $-U^{-1}S$) and find the LU factorization of S, $L_{22}U_{22} = S$.

   Then
   $$\begin{bmatrix} G+H & A^T \\ A & -U^{-1}S \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}.$$
   is the LU factorization of the *augmented system.* Since the system is symmetric and the matrix $G + H$ is positive definite, it would make more sense to perform the Choleski decomposition of $G + H$, but sparse LU decomposition seems to be much faster in MATLAB.

   This approach is beneficial when the matrix $G+H$ has some special structure, for example banded or block-diagonal and $n >> m$.

Remark 5.1 *Note that the above approach can be used in solving smooth convex quadratic problems with $n >> m$, since the blocks L_{11}, L_{21}, U_{11} and U_{12} have to be calculated only once. At each iteration, only $m \times m$ matrix S has to be factorized. MATLAB is 2-3 times faster than MOSEK on such QP examples.*

In Table 5.10, we summarize the timings for different ways of solving the linear system per iteration of IPM . The problem parameters are $M = 101$, $\epsilon = 0.0001$. In the case when both matrices are dense, we store them in a dense format. For

G \ A	dense			60%			40%			5%		
	Chol	Aug	Bcksl	Chol	Aug	Bcksl	Chol	Aug	Bcksl	Chol	Aug	Bcksl
dense	1.8	3.3	2.1	6.6	3.3	6.5	4.1	3.3	4	1.8	8.7	1.7
40%	12	6.3	10.2	6.8	5.3	6.6	4.2	10.4	4.1	1.8	3.1	1.8
5%	12	6.7	11.8	6.7	6.3	6.5	4.2	7.6	4.1	1.6	4.5	1.6

Table 5.10: MATLAB CPU times for different linear system solvers; n=1000,m=500

the Cholesky case, we do the matrix multiplication in (5.4) in sparse format, but then convert the matrix into dense format before performing the Cholesky decomposition. For the remaining two methods the data is kept in a sparse format.

For $\frac{n}{4} \leq m \leq \frac{n}{2}$, we found that whenever G and A were both full dense, CPU times for **Chol.** were half of those for **Aug.**. When we made A sparser (while keeping G full dense), the CPU times became equal around 40% density for A. When A had only 5% of its entries as non-zeroes, **Chol.** beat **Aug.** by a factor of five.

We notice that keeping the data in a sparse format is only beneficial when both matrices are very sparse, 5% or less. Otherwise, keeping the data in a dense format and doing the Cohleski decomposition is the fastest choice in MATLAB. We'll refer to this option as **Dense Chol.**.

Table 5.11 is created in a similar way. Only very sparse data is considered and the timing for the **Dense Chol.** option is given in a separate column. We can see that Cholesky is always faster than backslash. When G and A are $1-5\%$ sparse, Cholesky dominates all other methods. We notice that increasing the sparsity and decreasing the number of constraints improves the performance of the augmented system. When the sparsity is around 0.5% and the number of constraints is 10%

G \ A	5%			1%			
	Chol	Aug	Bcksl	Chol	Aug	Bcksl	Dense Chol
m=1000							
1%	22	26	33	13	12	34	33
0.5%	26	23	33	16	7	34	33
m=300							
1%	17	5	28	12	3	24	17
0.5%	16	3	28	12	1	19	17

Table 5.11: MATLAB CPU times for different linear system solvers; n=3000.

of the number of variables, the augmented system becomes the fastest choice.

In the next series of experiments we model real-life, large-scale portfolio optimization problems with thousands of variables. In such applications with very large n, a very sparse G, banded (or near-banded) matrix structure makes sense. For this set of experiments we generate G with overlapping blocks on the diagonal. We also add upper and lower bounds on all the variables. The number of constraints is equal to the number of blocks. We also add a budget constraint $x_1+x_2+\cdots+x_n \leq 1$. We noticed that addition of the bounds does not change the timings significantly. But the budget constraint makes the matrix of a condensed system dense. For these problems augmented system gives much better timings, so only **Aug.** and **Block LU** methods are represented in the Table 5.12.

	A					
	10%		5%		1%	
	Aug	Block LU	Aug	Block LU	Aug	Block LU
n=3000 (15 blocks)	2.1	1.7	1.6	1.5	0.6	1.2
n=6000 (30 blocks)	5.4	3.5	3.2	3.2	1.4	2.6
n=9000 (45 blocks)	10.6	5.6	5.6	5.1	2.4	4.0
n=12000 (60 blocks)	7.2	7.9	9.1	7.0	3.8	5.6

Table 5.12: MATLAB CPU times for different linear system solvers; block-diagonal G, blocks: 200×200, 10% dense; m=200; upper and lower bounds.

5.6 Experiments with Sparse Data

In this section we compare the timings for MOSEK and our IPM algorithm (in MATLAB) on sparse large scale data. The spline approximation parameter $\epsilon = 10e-5$. Both MOSEK and MATLAB were terminated when the relative gap was less than 10e-7. We noticed that for all the examples solved, the objective function $f(x)$ at the solutions given by MOSEK and MATLAB differ in the seventh or eights digit.

For the Tables 5.13, 5.14 and 5.15 matrix G was sparse but had no special structure. In Table 5.13 we solve the same problems changing only the number of the breakpoints. The timings of our method stay virtually unchanged , while the timings for the lifted problem solved in MOSEK increase. In the next series of tests we increase the dimension of the problem, G has 20 non-zeros per row, all the remaining parameters are fixed, $M = 25$. In this case our code beats MOSEK by a constant, see Table 5.14. For Table 5.15, we also increase the dimension of the problem, but keep the sparsity of G constant. In this case MOSEK performs better with the increase in dimension.

In the remaining tables the matrix G is block-diagonal with block size approximately 200×200. The blocks are overlapping by 10 diagonal elements on average. Each block is sparse.

As before, timings of our method stay virtually unchanged with increase in the number of breakpoints, while the timings for the lifted problem solved in MOSEK

Number of Breakpoints	MATLAB	MOSEK
$M = 101$	83 (15)	456 (13)
$M = 51$	78 (14)	230(12)
$M = 25$	82(15)	129 (12)
$M = 11$	80 (15)	70 (11)
$M = 3$	85 (15)	42 (10)
No Trans. Costs	74 (15)	30 (9)

Table 5.13: MATLAB and MOSEK CPU time (iterations) for different values of M; n=5000, G 0.5% dense; m=300, A 1% dense.

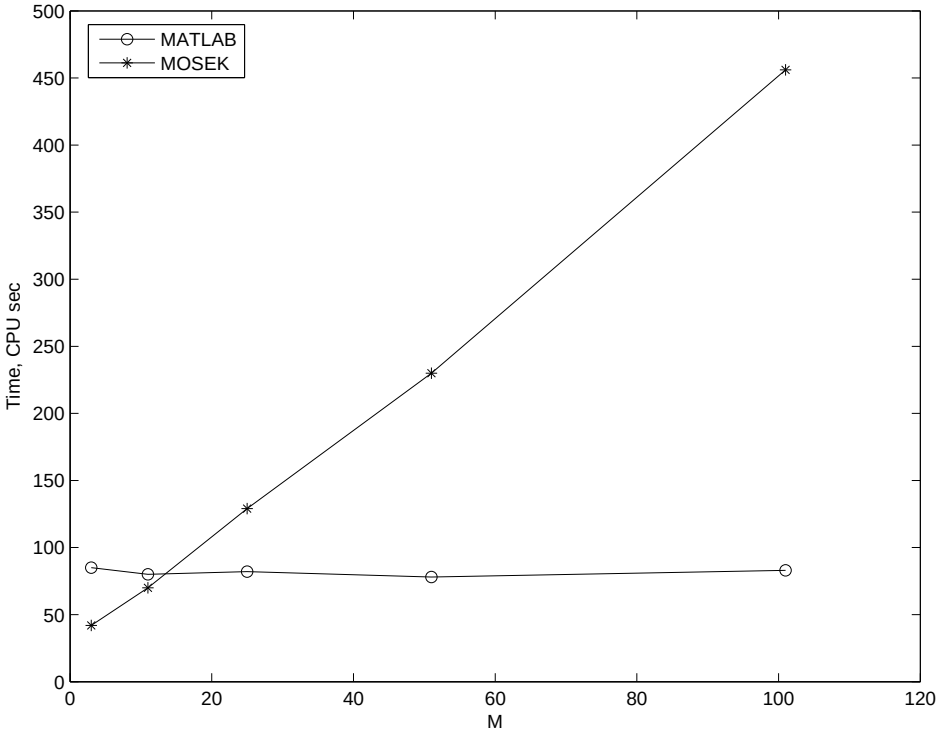

Figure 5.2: MATLAB and MOSEK CPU time (iterations) for different values of M; n=5000, G 0.5% dense; m=300, A 1% dense.

Dimension	MATLAB	MOSEK
n=21000	902 (13)	1000 (15)
n=18000	695 (14)	788 (15)
n=15000	433 (14)	588(15)
n=12000	262 (13)	370 (13)
n=9000	146 (13)	224 (11)
n=6000	71 (14)	143 (11)
n=3000	24 (14)	64 (11)

Table 5.14: MATLAB and MOSEK CPU time (iterations) for different values of n: G has 20 non-zeros per row; m=300, A 1% dense; M=25.

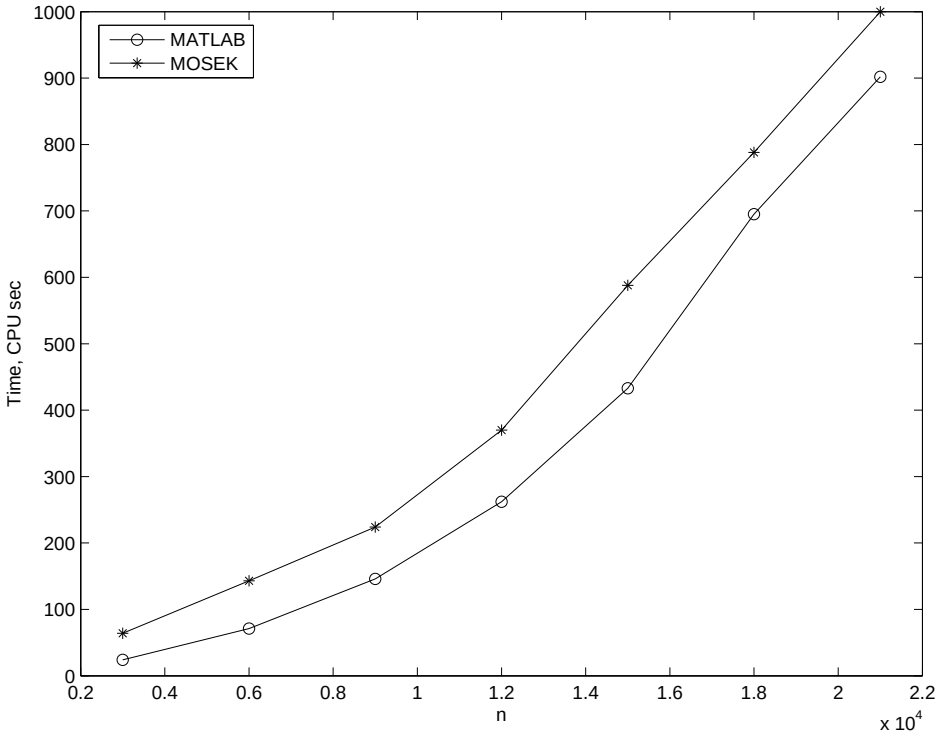

Figure 5.3: MATLAB and MOSEK CPU time (iterations) for different values of n: G has 20 non-zeros per row; m=300, A 1% dense; M=25.

Dimension	MATLAB	MOSEK
n=12000	1980 (13)	1026(11)
n=9000	593(14)	425 (11)
n=6000	117 (13)	162 (11)
n=3000	16 (13)	63 (11)

Table 5.15: MATLAB and MOSEK CPU time (iterations) for different values of n: G is 0.5% dense; m=300, A 1% dense; M=25.

increase, Table 5.16. For Table 5.17, we increase the dimension of the problem, but keep the block size constant. In this case our MATLAB code beats MOSEK by a constant factor. Also note that MOSEK is approximately 2 times faster on a smooth problem without the transaction costs than our MATLAB code.

Some additional experiments on a very large data are reported in Table 5.18. Note that for these problems MOSEK spends around 50% of time on preprocessing.

Number of Breakpoints	MATLAB	MOSEK
$M = 101$	97 (13)	825 (13)
$M = 51$	95 (13)	440 (13)
$M = 25$	94 (13)	215 (11)
$M = 11$	95 (13)	117 (10)
$M = 3$	101 (14)	78 (10)
No Trans. Costs	93 (13)	46 (9)

Table 5.16: MATLAB and MOSEK CPU time (iterations) for different values of M: block-diagonal G, 45 blocks 200×200, 10% dense; m=200, A 10% dense; upper and lower bounds.

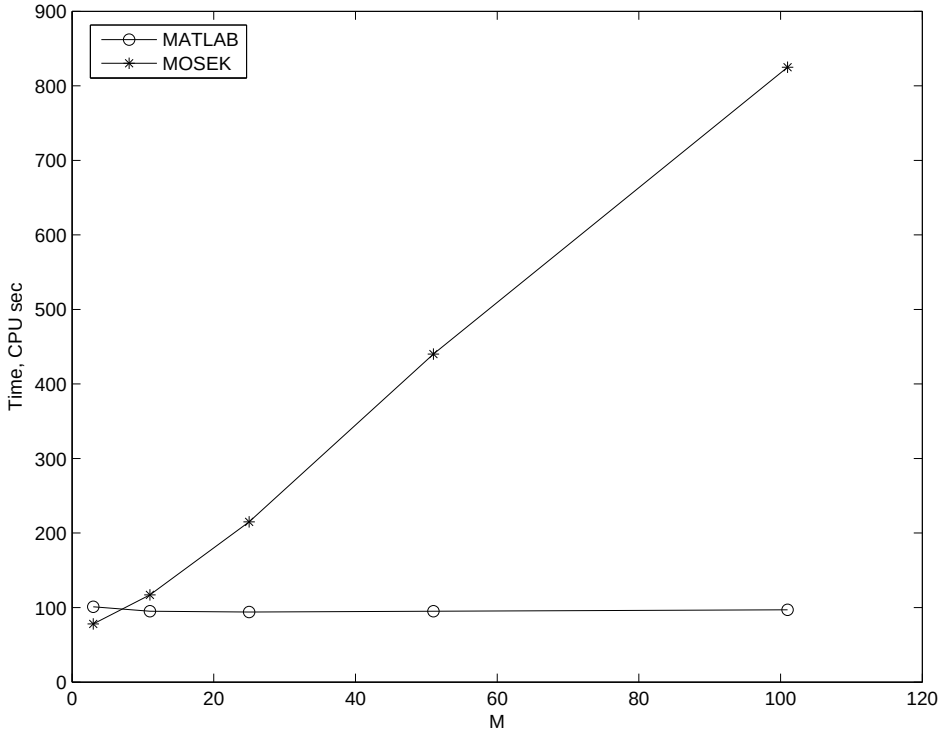

Figure 5.4: MATLAB and MOSEK CPU time (iterations) for different values of M: block-diagonal G, 45 blocks 200×200, 10% dense; m=200, A 10% dense; upper and lower bounds.

Number of Blocks	MATLAB	MOSEK
75 blocks n=15000	164 (13)	401 (11)
60 blocks n=12000	131 (13)	303(11)
45 blocks n=9000	94 (13)	215 (11)
30 blocks n=6000	53 (12)	135 (11)
15 blocks n=3000	26 (12)	64 (11)

Table 5.17: MATLAB and MOSEK CPU time (iterations) for different number of blocks: block-diagonal G, blocks: 200×200, 10% dense; m=200, A 10% dense; upper and lower bounds, M=25.

n	Blocks				A		M	MATLAB	MOSEK
	Number	Size	Density	Overlap	m	Density			
53400	89	600	0.006	9	500	0.1	51	3114 (15)	8797(11)
100000	1000	100	0.1	10	200	0.01	25	2966(15)	5595(16)
200000	10000	20	0.1	5	300	0.01	11	18010 (17)	CAN'T SOLVE

Table 5.18: MATLAB and MOSEK CPU time (iterations) for some large-scale problems.

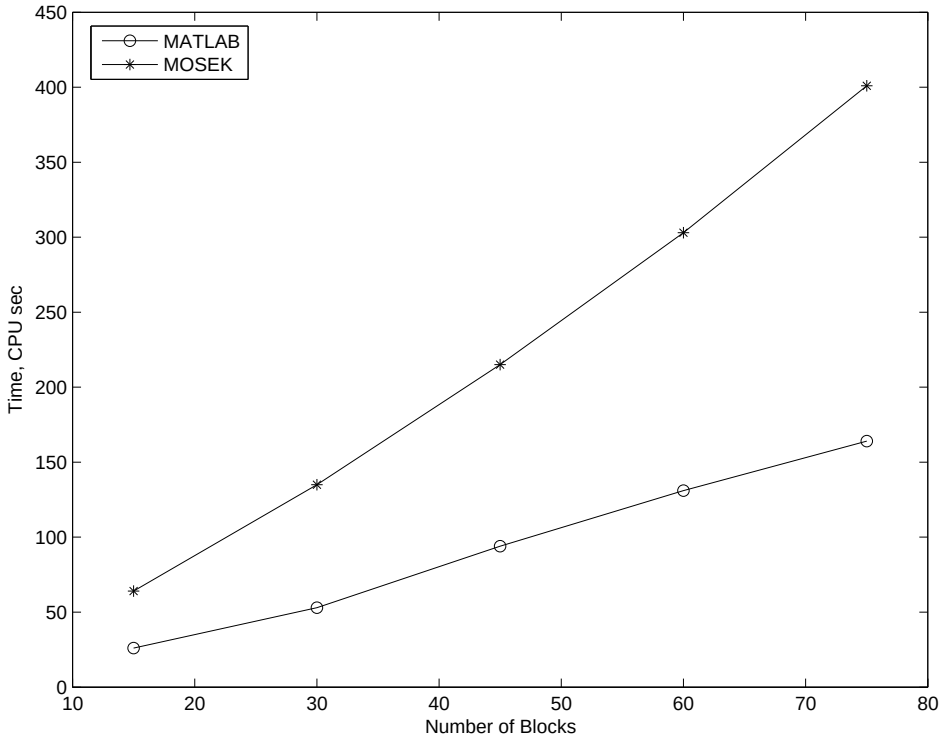

Figure 5.5: MATLAB and MOSEK CPU time (iterations) for different values of n: block-diagonal G, blocks: 200×200, 10% dense; m=200, A 10% dense; upper and lower bounds, M=25.

Chapter 6

Conclusion

In this thesis we considered the expected utility maximization problem in presence of convex non-differentiable transaction costs and linear or piece-wise linear constraints. We used subdifferentials to derive the optimality conditions for this problem, which allowed us to develop the relevant theory in a clear and concise way.

We showed that the problem can be solved efficiently without introducing new variables or constraints. Two major approaches were considered.

We showed that approximating the transaction costs with spline functions and solving the smooth problem with the interior point methods gives a very good approximation to the optimal solution.

We also developed an active set algorithm for solving the above problem and showed that under some assumptions standard for the active set method this algorithm terminates in a finite number of steps. This algorithm performs best when a "warm start" is possible.

There are some natural extensions to the results of this thesis.

It seems that choosing the right parameters for the implementation of the interior point method can significantly improve it's performance in the case when the objective function becomes "less smooth". Another possible way to speed up the IPM is varying the spline parameter ϵ from iteration to iteration.

It would be interesting to find an efficient measure of smoothness for such functions and give better estimates of the number of the coordinates of the optimal solution coinciding with the breakpoints.

One can try to extend Algorithm 2 to the case of non-differentiable nonconvex transaction costs. Many of the existing algorithms for nonconvex optimization are based on optimizing a sequence of convex approximations. Using our algorithm for solving these subproblems would speed up the process and could be a valuable source of heuristic ideas.

Another direction for further exploration is how degeneracy can be resolved in our algorithm. It seems possible to generalize some of the existing methods, for example Bland's rule for linear programming, to the non-differentiable case.

Finally, it would be very interesting to investigate in more detail the effect of transaction costs on the choice of the optimal portfolio, both numerically and analytically.

Appendix A

Equivalent Problems

Recall the problem (1.6).

$$\begin{aligned}
\min \quad & F(x) + \sum_{i=1}^{n} \sum_{l=0}^{M_i^+} f_{il}^+(x_{il}^+) + \sum_{i=1}^{n} \sum_{l=0}^{M_i^-} f_{il}^-(x_{il}^-) \\
s.t. \quad & Ax \leq b, \\
& x_i - \sum_{i=1}^{n} \sum_{l=0}^{M_i^+} x_{il}^+ + \sum_{i=1}^{n} \sum_{l=0}^{M_i^-} x_{il}^- = \hat{x}_i, \ \text{for } i = 1, ..., n, \\
& 0 \leq x_{il}^+ \leq d_{il+1}^+, \ \text{for } i = 1, ..., n, \ l = 0, ..., M_i^+, \\
& 0 \leq x_{il}^- \leq d_{il+1}^-, \ \text{for } i = 1, ..., n, \ l = 0, ..., M_i^-.
\end{aligned}$$

Let us denote the objective function of this problem by $\bar{f}(x, x^-, x^+)$.

Lemma A.1 *Problem (2.1) is equivalent to the problem (1.6) in the sense that*

(i) x is optimal for (2.1) if and only if (x, x^-, x^+) is optimal for (1.6) for some (x^-, x^+) and

$$f(x) = \bar{f}(x, x^-, x^+);$$

(ii) (2.1) is unbounded if and only if (1.6) is unbounded.

Proof. Suppose that x is feasible for (2.1). Then we can form the vector (x, x^m, x^p) form x in the following way:

for each index i such that $x_i \geq \hat{x}_i$ and $x_i - \hat{x}_i \in [d^+_{iL}, d^+_{iL+1}]$

$$\begin{aligned}
&x_i^{lm} = 0, && \text{for} \quad l = 1, .., M_i^-, \\
&x_i^{lp} = d^+_{il}, && \text{for} \quad l = 1, .., L-1, \\
&x_i^{Lp} = x_i - \hat{x}_i - d^+_{iL}, && \\
&x_i^{lp} = 0, && \text{for} \quad l = L+1, .., M_i^+,
\end{aligned}$$

and for each index i such that $x_i < \hat{x}_i$ and $\hat{x}_i - x_i \in [d^-_{iL}, d^-_{iL+1}]$

$$\begin{aligned}
&x_i^{lm} = 0, && \text{for} \quad l = L+1, .., M_i^- \\
&x_i^{Lm} = \hat{x}_i - x_i - d^-_{iL}, && \\
&x_i^{lm} = d^-_{il}, && \text{for} \quad l = 1, .., L-1, \\
&x_i^{lp} = 0, && \text{for} \quad l = 1, .., M_i^+.
\end{aligned}$$

By construction this vector is feasible for (1.6) and objective functions of these problems will coincide on x and (x, x^m, x^p) respectively:

$$f(x) = \bar{f}(x, x^m, x^p).$$

Suppose a vector (x, x^-, x^+) is feasible for (1.6) but vector (x^-, x^+) is different from (x^m, x^p). Then at least one of the cases described below applies:

Case 1. Suppose $x^+_{ik} < d^+_{ik+1}$ and $x^+_{ij} > 0$ for some $k < j$. Then we can shift $\epsilon = \min\{d^+_{ik+1} - x^+_{ik}, x^+_{ij}\}$ from x^+_{ij} to x^+_{ik}. The resulting vector will remain feasible. Below we use the convexity of f^+_{ik} and f^+_{ij} and the fact that

$$\frac{\partial f^+_{ik}}{\partial x^+_{ik}}(x^+_{ik} - \epsilon) \leq \frac{\partial f^+_{ik}}{\partial x^+_{ik}}(d_{ik+1}) \leq \frac{\partial f^+_{ij}}{\partial x^+_{ij}}(d_{ij}) \leq \frac{\partial f^+_{ij}}{\partial x^+_{ij}}(x^+_{ij} + \epsilon)$$

to show that the value of the objective value will decrease after the shift. Only two terms of the objective function will be affected by this change:

$$\begin{aligned} & f_{ik}^{+}(x_{ik}^{+}) + f_{ij}^{+}(x_{ij}^{+}) \geq \\ f_{ik}^{+}(x_{ik}^{+}+\epsilon) - \epsilon\frac{\partial f_{ik}^{+}}{\partial x_{ik}^{+}}(x_{ik}^{+}+\epsilon) + f_{ij}^{+}(x_{ij}^{+}-\epsilon) + \epsilon\frac{\partial f_{ij}^{+}}{\partial x_{ij}^{+}}(x_{ij}^{+}-\epsilon) \geq \\ f_{ik}^{+}(x_{ik}^{+}+\epsilon) + f_{ij}^{+}(x_{ij}^{+}-\epsilon) + \epsilon\left(\frac{\partial f_{ij}^{+}}{\partial x_{ij}^{+}}(x_{ij}^{+}-\epsilon) - \frac{\partial f_{ik}^{+}}{\partial x_{ik}^{+}}(x_{ik}^{+}+\epsilon)\right) \geq \\ & f_{ik}^{+}(x_{ik}^{+}+\epsilon) + f_{ij}^{+}(x_{ij}^{+}-\epsilon). \end{aligned}$$

After a finite number of such steps Case 1 will no longer occur.

Case 2. $x_{ik}^{-} < d_{ik+1}^{-}$ and $x_{ij}^{-} > 0$ for some $k < j$. Then we can shift $\epsilon = \min\{d_{ik+1}^{-} - x_{ik}^{-}, x_{ij}^{-}\}$ from x_{ij}^{-} to x_{ik}^{-} and the resulting vector will remain feasible. Similar to the Case 1, the value of the objective value will decrease after the shift:

$$\begin{aligned} & f_{ik}^{-}(x_{ik}^{-}) + f_{ij}^{-}(x_{ij}^{-}) \geq \\ f_{ik}^{-}(x_{ik}^{-}+\epsilon) - \epsilon\frac{\partial f_{ik}^{-}}{\partial x_{ik}^{-}}(x_{ik}^{-}+\epsilon) + f_{ij}^{-}(x_{ij}^{-}-\epsilon) + \epsilon\frac{\partial f_{ij}^{-}}{\partial x_{ij}^{-}}(x_{ij}^{-}-\epsilon) \geq \\ f_{ik}^{-}(x_{ik}^{-}+\epsilon) + f_{ij}^{-}(x_{ij}^{-}-\epsilon) + \epsilon\left(\frac{\partial f_{ij}^{-}}{\partial x_{ij}^{-}}(x_{ij}^{-}-\epsilon) - \frac{\partial f_{ik}^{-}}{\partial x_{ik}^{-}}(x_{ik}^{-}+\epsilon)\right) \geq \\ & f_{ik}^{-}(x_{ik}^{-}+\epsilon) + f_{ij}^{-}(x_{ij}^{-}-\epsilon). \end{aligned}$$

After a finite number of such steps Case 2 will no longer occur.

Case 3. $x_{ik}^{+} > 0$ and $x_{ij}^{-} > 0$ for some k and j. Then we can deduct $\epsilon = \min\{x_{ik}^{+}, x_{ij}^{-}\}$ from both x_{ik}^{+} and x_{ij}^{-} and the resulting vector will remain feasible. Only two terms of the objective functions will be affected by this change. Since f_{ik}^{+} and f_{ij}^{+} are increasing functions,

$$\begin{aligned} & f_{ik}^{+}(x_{ik}^{+}) + f_{ij}^{-}(x_{ij}^{-}) \geq \\ & f_{ik}^{+}(x_{ik}^{+}-\epsilon) + f_{ij}^{-}(x_{ij}^{-}-\epsilon). \end{aligned}$$

After a finite number of such steps Case 3 will no longer occur.

If none of the above 3 cases apply, the resulting vector is equal to (x, x^m, x^p) and therefore

$$\bar{f}(x, x^-, x^+) \geq \bar{f}(x, x^m, x^p) = f(x).$$

(i) Suppose that x is optimal for (2.1). Then (x, x^m, x^p) is feasible for (1.6) and for any feasible (y, y^-, y^+)

$$\bar{f}(y, y^-, y^+) \geq \bar{f}(y, y^m, y^p) = f(y) \geq f(x) = \bar{f}(x, x^m, x^p).$$

Therefore (x, x^m, x^p) is optimal for (1.6).

Suppose that (x, x^-, x^+) is optimal for (1.6). Then x is feasible for (2.1). Suppose $y \neq x$ is optimal for (2.1). Then

$$\bar{f}(y, y^m, y^p) = f(y) < f(x) = \bar{f}(x, x^m, x^p) \leq \bar{f}(x, x^-, x^+).$$

This contradiction proves that x is optimal for (2.1).

(ii) Suppose that (2.1) is unbounded. Then there exists a sequence of feasible points $\{x_k\}$ such that $\lim_{k\to\infty} f(x_k) = -\infty$. Then the points $\{(x_k, x_k^m, x_k^p)\}$ are feasible for (1.6) and

$$\lim_{k\to\infty} \bar{f}(x_k, x_k^m, x_k^p) = \lim_{k\to\infty} f(x_k) = -\infty$$

. Therefore (1.6) is also unbounded.

Suppose that (1.6) is unbounded, i.e. there exists a sequence of the feasible points $\{(x_k, x_k^-, x_k^+)\}$ such that $\lim_{k\to\infty} \bar{f}(x_k, x_k^-, x_k^+) = -\infty$. Then the points $\{x_k\}$ are feasible for(2.1) and

$$\lim_{k\to\infty} f(x_k) = \lim_{k\to\infty} \bar{f}(x_k, x_k^m, x_k^p) \leq \lim_{k\to\infty} \bar{f}(x_k, x_k^-, x_k^+) = -\infty.$$

Hence (2.1) is also unbounded. □

Lemma 3.1 *Let the function g_k be a piecewise linear convex function defined by (3.2). Then*

$$g_i^k(x) = \max_{l=0,\ldots,M_i} a_{il}^k x + b_{il}^k.$$

Moreover,

(i) for any $l = 1, \ldots, M_i$, for any x satisfying $d_{il} < x < d_{il+1}$,

$$g_i^k(x) = a_{ij}^k x + b_{ij}^k$$

if an only if $a_{ij}^k = a_{il}^k$;

(ii) for any $l = 1, \ldots, M_i$,

$$g_i^k(d_{il}) = a_{ij}^k d_{il} + b_{ij}^k$$

if an only if $a_{ij}^k = a_{il}^k$ or $a_{ij}^k = a_{il-1}^k$.

Proof. By Proposition B.1,

$$a_1^k \leq a_2^k \leq \cdots \leq a_{M_i}^k.$$

For any l, there are two possibilities :

(i) $a_{il}^k = a_{il+1}^k$. Note that the linear functions $a_{il}^k x + b_{il}^k$ and $a_{il+1}^k x + b_{il+1}^k$ intersect at d_{il+1}, therefore they coincide. We get that $b_i^k l = b_{il+1}^k$ and $a_{il}^k x + b_{il}^k = a_{il+1}^k x + b_{il+1}^k$ for any $x \in \mathbb{R}^n$.

(ii) $a_{il}^k < a_{il+1}^k$. Then

$$a_{il}^k x + b_{il}^k > a_{il+1}^k x + b_{il+1}^k$$

if and only if

$$x < \frac{b_{il}^k - b_{il+1}^k}{a_{il+1}^k - a_{il}^k} = d_{il+1}.$$

Therefore for any x satisfying $d_{il} < x < d_{il+1}$,

$$a_{i0}^k x + b_{i0}^k \leq \cdots \leq a_{il-1}^k x + b_{il-1}^k \leq a_{il}^k x + b_{il}^k \geq a_{il+1}^k x + b_{il+1}^k \geq \cdots \geq a_{iM_i}^k x + b_{iM_i}^k.$$

The $j-th$ inequality is satisfied with equality if and only if $a_{ij-1}^k = a_{ij}^k$.

Next, suppose that $x = d_{il}$, then

$$a_{i0}^k x + b_{i0}^k \leq \cdots \leq a_{il-1}^k x + b_{il-1}^k = a_{il}^k x + b_{il}^k \geq a_{il+1}^k x + b_{il+1}^k \geq \cdots \geq a_{iM_i}^k x + b_{iM_i}^k.$$

Note that the terms number l and $l-1$ are always equal. The $j-th$ inequality, where $j \neq l$, is satisfied with equality if and only if $a_{ij-1}^k = a_{ij}^k$. □

Lemma 3.2 *Consider the problem*

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & \sum_{i=1}^n y_i^k \leq h^k, \;\; k = 1, \ldots, m, \\ & y_i^k \geq a_{il}^k x_i + b_{il}^k \;\; i = 1, \ldots, n, \;\; k = 1, \ldots, m. \end{array} \tag{A.1}$$

Then $x^* \in \mathbb{R}^n$ is optimal for (3.1) if and only if (x^*, y^*) is optimal for (3.5) for some $y^* \in \mathbb{R}^{nm}$.

Proof. Let us denote the feasible region of the problem (3.1) by S_1 and denote the set of the points $x \in \mathbb{R}^n$, such that (x, y) is feasible for (3.5) for some $y \in \mathbb{R}^{nm}$ by S_2. since the objective function of the problem (3.5) depends only on x, solving this problem is equivalent to minimizing $f(x)$ over S_2. If we show that $S_1 = S_2$, the result will be proven.

Suppose that x is feasible for (3.1). Then we can form the vector (x, y) from x in the following way:

$$y_i^k = \max\{a_{ij}^k x_i^* + b_{ij}^k, \;\; j = 1, \ldots, M_i\}, \text{ for each } i = 1, 2, \ldots, n.$$

Clearly,

$$y_i^k \geq a_{ij}^k x_i^* + b_{ij}^k, \text{ for each } \quad j = 1, \ldots, M_i, \text{ for each } i = 1, 2, \ldots, n.$$

By Lemma 3.1, $y_i^k = g_i^k(x)$ and $\sum_{i=1}^n y_i^k = \sum_{i=1}^n g_i^k(x) \leq h^k$ from feasibility of x. Vector (x, y) is feasible for (3.5). Therefore $S_1 \subset S_2$.

Next, suppose that (x, y) is feasible for (3.5). Then, using Lemma 3.1,

$$\sum_{i=1}^n g_i^k(x) = \sum_{i=1}^n \max_{j=1,\ldots,M_i} \{a_{ij}^k x_i^* + b_{ij}^k\} \leq \sum_{i=1}^n y_i^k \leq h_k.$$

This proves that x is feasible for (3.1). Therefore $S_2 \subset S_1$.

We showed that $S_1 = S_2$. This proves the result. □

Appendix B

Optimality Conditions for the Subproblems

The following proposition provides necessary and sufficient conditions for the convexity of the function $f_i(x_i)$ given by (2.3).

Proposition B.1 *Assume that f_{il} is convex for all $l = 0, \ldots, M_i$. Then the function $f_i(x_i)$ defined by (2.3) is convex if and only if*

$$f_{il-1}(d_{il}) = f_{il}(d_{il}) \text{ for each } \quad l = 1, \ldots, M_i \tag{B.1}$$

and

$$\frac{df_{il-1}^{-}}{dx_i}(d_{il}) \leq \frac{df_{il}^{+}}{dx_i}(d_{il}) \text{ for each } \quad l = 1, \ldots, M_i.^{1} \tag{B.2}$$

Proof. First, suppose $f_i(x_i)$ is convex on $\mathbb{R}$. Since convexity implies continuity, equations (B.1) are satisfied.

[1] In the above df^- and df^+ denote the left and the right derivatives of f respectively.

It is well known (see [29] for example), that if $\theta(t)$ is a convex function , then is right-differentiable and left-differentiable and for any $t \in \mathbb{R}$

$$\frac{d\theta^-}{dt}(t) \leq \frac{d\theta^+}{dt}(t).$$

So, for any $l = 1, \ldots, M_i$ we have

$$\frac{df_{il-1}^-}{dt}(d_{il}) \leq \frac{df_{il}^+}{dt}(d_{il}).$$

To prove the converse, we note that the $f_i(x_i)$ is continuous, right-deffirentiable and the right derivative $\frac{df_{il}^+}{dx_i}$ is increasing. This implies that $f_i(x_i)$ is convex, (see [29]). □

We formulate the optimality conditions for the subproblems that have to be solved at each iteration of the Modified Algorithm 2.

Let us consider a problem of minimizing function $f(x)$ given by (2.2), (2.3) over a region B,

$$B = \left\{ x \;\middle|\; \begin{array}{ll} g^k(x) \leq h^k, & k = 1, \ldots, m, \\ d_{il} \leq x_i \leq d_{il+1}, & i \in J, \text{ for some } l \in \{0, ..., M_i\}, \\ x_i = d_{il}, & i \notin J, \text{ for some } l \in \{1, ..., M_i\}, \end{array} \right\} \tag{B.3}$$

where J is a subset of $\{1, 2, ...n\}$.

Because of the structure of the feasible region, each of the piece-wise linear inequality constraints can be substituted by a single linear constraint. The separable part of the objective function can also be simplified.

For each $x \in B$, if $i \in J$ and $d_{il} \leq x_i \leq d_{il+1}$ or $d_{il} \leq x_i$ or $x_i \leq d_{il+1}$ is a corresponding constraint, then $g_i^k(x_i) = a_{il}^k + b_{il}^k$ and $f_i(x_i) = f_{il}(x_i)$. If $i \notin J$ and $x_i = d_{il}$ is a corresponding constraint, then $g_i^k(x_i) = a_{il-1}^k + b_{il-1}^k = a_{il}^k + b_{il}^k$ and $f_i(x_i) = f_{il}(x_i) = f_{il-1}(x_i)$.

Therefore, the region B can can be represented by

$$B = \left\{ x \;\middle|\; \begin{array}{lll} \sum_{i\in J}(a_{il}^k x_i + b_{il}^k) + & \sum_{i\notin J}(a_{i\bar{L}}^k x_i + b_{i\bar{L}}^k) \le h^k, & k = 1,\ldots,m, \\ d_{il} \le x_i \le d_{il+1}, i \in J, & \text{for some} & l \in \{0,...,M_i\}, \\ x_i = d_{il}, i \notin J, & \text{for some} & l \in \{1,...,M_i\}, \end{array} \right\} \quad \text{(B.4)}$$

where, for each $i \notin J$ with corresponding constraint $x_i = d_{il}$, $\bar{L}$ can be either l or $l-1$.

Lemma B.1 *Let $f(x)$ be defined by (2.2), (2.3) and let the region B be defined by (B.4). For any $x \in B$, the objective function $f(x)$ satisfies*

$$f(x) = F(x) + \sum_{i\in J} f_{il}(x_i) + \sum_{i\notin J} f_{iL}(x_i), \quad \text{(B.5)}$$

where, for each $i \notin J$ with corresponding constraint $x_i = d_{il}$, L can be either l or $l-1$.

Since the above two functions coincide on B, minimizing $f(x)$ over B is equivalent to minimizing the right hand side of (B.5) over B. We summarize this result in the following lemma

Lemma B.2 *Let $f(x)$ be defined by (2.2), (2.3) and let the region B be defined by (B.4). Then x minimizes $f(x)$ over B if and only if x minimizes*

$$F(x) + \sum_{i\in J} f_{il}(x_i) + \sum_{i\notin J} f_{iL}(x_i) \quad \text{(B.6)}$$

over B, where, for each $i \in J$, l is such that $d_{il} \le x_i \le d_{il+1}$ or $d_{il} \le x_i$ or $x_i \le d_{il+1}$ is a corresponding constraint of B, for each $i \notin J$, L is such that either $x_i = d_{iL}$ or $x_i = d_{iL+1}$ is a corresponding constraint of B.

We can now state the optimality conditions for our subproblem:

By Lemma B.2, we can just give the optimality conditions for a function of the form (B.6), which is convex and twice differentiable. Let us denote by A the matrix formed by the coefficients of the linear constraints a_{il}^k and let us denote by b the vector of the right-hand sides of these constraints. Then $x \in \mathbb{R}^n$ minimizes $f(x)$ over B if and only if

$$\left.\begin{array}{lll}
Ax \leq b, & & \\
d_{il-1} \leq x_i \leq d_{il}, & \text{for all } i \in J, & \text{for some } l \in \{1, ..., M_i\}, \\
x_i = d_{il}, & \text{for all } i \notin J, & \text{for some } l \in \{1, ..., M_i\}, \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & = (A^T u)_i + r_i - s_i, & \text{for all } i \in J \\
 & & \text{with } x_i \in [d_{il}, d_{il+1}], \\
-(\nabla F(x))_i - \frac{df_{iL}}{dx_i}(x_i) & = (A^T u)_i + \lambda_i, & \text{for all } i \notin J \text{with} \\
 & & x_i = d_{il},\ L \in \{l-1, l\}, \\
u \geq 0, & & \\
u^T(Ax - b) = 0. & & \\
s_i \geq 0 & & \text{for all } i \in J, \\
r_i \geq 0 & & \text{for all } i \in J, \\
s_i(x_{il} - d_{il}) = 0 & \text{for all } i \in J & \text{with } x_i \in [d_{il}, d_{il+1}], \\
r_i(d_{il+1} - x_{il}) = 0 & \text{for all } i \in J & \text{with } x_i \in [d_{il}, d_{il+1}].
\end{array}\right\}$$

We first observe that the set of equalities corresponding to $i \notin J$ is only needed to define λ_i, and there are no other equalities or inequalities containing λ_i. Therefore we can drop these equations.

Note that, if $i \in N(x)$, then from the complementary slackness

$s_i = r_i = 0.$

If $J \cap E(x)$ is not empty, then for each $i \in J \cap E(x)$ at least one of s_i and r_i is zero. This allows us to rewrite the optimality conditions in the following way:

$$\left.\begin{array}{lll}
Ax \leq b, & & \\
d_{il-1} \leq x_i \leq d_{il}, & \text{for all } i \in J, & \text{for some } l \in \{1, ..., M_i\}, \\
x_i = d_{il}, & \text{for all } i \notin J, & \text{for some } l \in \{1, ..., M_i\}, \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & = (A^T u)_i, & \text{for all } i \in N(x) \\
 & & \text{with } x_i \in (d_{il}, d_{il+1}), \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & = (A^T u)_i + r_i, & \text{for all } i \in J \cap E(x) \\
r_i \geq 0, & & \text{with } x_i = d_{il+1}, \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & = (A^T u)_i - s_i, & \text{for all } i \in J \cap E(x) \\
s_i \geq 0, & & \text{with } x_i = d_{il}, \\
u \geq 0, & & \\
u^T(Ax - b) = 0. & &
\end{array}\right\}$$

or

$$\left.\begin{array}{lll}
Ax \leq b, & & \\
d_{il-1} \leq x_i \leq d_{il}, & \text{for all } i \in J, & \text{for some } l \in \{1, ..., M_i\}, \\
x_i = d_{il}, & \text{for all } i \notin J, & \text{for some } l \in \{1, ..., M_i\}, \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & = (A^T u)_i, & \text{for all } i \in N(x), \\
 & & \text{with } x_i \in (d_{il}, d_{il+1}), \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & \geq (A^T u)_i, & \text{for all } i \in J \cap E(x), \\
 & & \text{with } x_i = d_{il+1}, \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & \leq (A^T u)_i, & \text{for all } i \in J \cap E(x), \\
 & & \text{with } x_i = d_{il}, \\
u \geq 0, & & \\
u^T(Ax - b) = 0. & &
\end{array}\right\}$$

Another way to rewrite these conditions is

$$\left.\begin{array}{lll}
Ax \le b, & & \\
d_{il-1} \le x_i \le d_{il}, & \text{for all } i \in J, & \text{for some } l \in \{1, ..., M_i\}, \\
x_i = d_{il}, & \text{for all } i \notin J, & \text{for some } l \in \{1, ..., M_i\}, \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & = (A^T u)_i, & \text{for all } i \in N(x) \\
 & & \text{with } x_i \in (d_{il}, d_{il+1}), \\
-(\nabla F(x))_i - \frac{df_{il-1}}{dx_i}(x_i) & \ge (A^T u)_i, & \text{for all } i \in J \cap E(x) \\
 & \text{with } x_i = d_{il}, & \text{where } x_i \text{ is at an upper bound}, \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & \le (A^T u)_i, & \text{for all } i \in J \cap E(x) \\
 & \text{with } x_i = d_{il}, & \text{where } x_i \text{ is at a lower bound}, \\
u \ge 0, & & \\
u^T(Ax - b) = 0. & &
\end{array}\right\} \quad \text{(B.7)}$$

Recall that for each $i \in J$ where $d_{il} \le x_i \le d_{il+1}$ is a corresponding constraint, $A_{ki} = a_{il}^k$ and therefore

$$\left.\begin{array}{lll}
Ax \le b, & & \\
d_{il-1} \le x_i \le d_{il}, & \text{for all } i \in J, & \text{for some } l \in \{1, ..., M_i\}, \\
x_i = d_{il}, & \text{for all } i \notin J, & \text{for some } l \in \{1, ..., M_i\}, \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & = \sum_{k=1}^m a_{il}^k u_k, & \text{for all } i \in N(x) \\
 & & \text{with } x_i \in (d_{il}, d_{il+1}), \\
-(\nabla F(x))_i - \frac{df_{il-1}}{dx_i}(x_i) & \ge \sum_{k=1}^m a_{il}^k u_k, & \text{for all } i \in J \cap E(x) \\
 & \text{with } x_i = d_{il}, & \text{where } x_i \text{ is at an upper bound}, \\
-(\nabla F(x))_i - \frac{df_{il}}{dx_i}(x_i) & \le \sum_{k=1}^m a_{il}^k u_k, & \text{for all } i \in J \cap E(x) \\
 & \text{with } x_i = d_{il}, & \text{where } x_i \text{ is at a lower bound}, \\
u \ge 0, & & \\
u^T(Ax - b) = 0. & &
\end{array}\right\} \quad \text{(B.8)}$$

We summarize this in the following Lemma.

Lemma B.3 *Let the function $f(x)$ be defined by (2.2), (2.3) and let the region B be defined by (B.3). Then x minimizes $f(x)$ over B if and only if (B.8) are satisfied.*

Appendix C

Linear Independence

Assume that Algorithm 2 is applied to problem (3.1). Let the vectors α_k^j be defined by the j iteration of the Algorithm and let $I \subseteq \{1, \ldots, m\}$, $E \subseteq \{1, \ldots, n\}$. Assume further that the vectors

$$\{\alpha_k^j,\ k \in I,\ e_i,\ i \in E\} \tag{C.1}$$

are linearly independent.

Let us assume without the loss of generality that $E = \{1, 2, \ldots, p\}$. Therefore $|E| = p$. Let us denote by A the matrix formed by the vectors α_k^j, $k = 1, \ldots, m$, denote by A_E the submatrix of A^E formed by the it's first p columns, and denote by A_N the submatrix of A formed by the remaining $n - p$ columns.

Then the matrix formed by the vectors (C.1) has the following form

$$\left[\begin{array}{c|c} A_E & A_N \\ \hline I & 0 \end{array}\right].$$

Note that this matrix is nonsingular if and only if the total number of the constraints in I plus the number of the elements in E is less or equal than n and the matrix A_N is non-singular.

Lemma C.1 *Vectors (C.1) are linearly independent if and only if the total number of the constraints in I plus the number of the elements in E is less or equal than n and the matrix A_N is non-singular.*

Recall that we refer to a vector $x \in R^n$ as **non-degenerate** for the problem (3.1) if the vectors

$$\{\alpha_k^j,\ k \in I(x^j),\ e_i,\ i \in E(x^j)\} \tag{C.2}$$

are linearly independent.

However the choice of α_k^j for a given point x^j is not unique.

Lemma C.2 *Let x^j be feasible for the problem (3.1). Let the matrix A_N formed as follows: for each $i \in E(x^j)$, $k \in I(x^j)$*

$$(A_N)_i^k = a_{il}^k,\ \text{where } l = D_i(x^j). \tag{C.3}$$

Then then the vectors (C.2) are linearly independent if and only if the total number of the constraints in $I(x^j)$ plus the number of the elements in $E(x^j)$ is less or equal than n and the matrix A_N is non-singular.

This shows that our definition of a non-degenerate point is consistent.

Bibliography

[1] L. ALTANGEREL, R.I. BOŢ, and G. WANKA. Duality for convex partially separable optimization problems. *Mong. Math. J.*, 7:1–18, 2003.

[2] A. BEN-TAL, A. NEMIROVSKI. *Lectures on Modern Convex Optimization, Analysis, Algorithms, and Engineering Applications.* MPS-SIAM series on Optimization, 2001.

[3] A. BEN-TAL, A. NEMIROVSKI. *Robust Convex Optimization.* Mathematics of Operations Research, vol. 23, pp.769-805, 1998.

[4] D.P. BERTSEKAS. Nonlinear Optimization. Athena Scientific, Belmont, 1999.

[5] DIMITRIS BERTSIMAS, CHRISTOPHER DARNELL, and ROBERT SOUCY. *Portfolio construction through mixed-integer programming at Grantham.* Mayo, Van Otterloo and Company, Interfaces, vol.29, pp.49-66, 1999.

[6] MICHAEL J. BEST and JAROSLAVA HLOUSKOVA. *An Algorithm for Portfolio Optimization with Transaction Costs.* Management Science, vol.51, No.11, pp.1676-1688, 2005.

[7] MICHAEL J. BEST and JAROSLAVA HLOUSKOVA. *An Algorithm for Portfolio Optimization with Variable Transaction Costs.* Combinatorics and Optimization Research Report 2004-32, December 2004.

[8] MICHAEL J. BEST and JAROSLAVA HLOUSKOVA. *Portfolio selection and transaction costs.* Computational Optimization and Applications, vol. 24, pp.95-116, 2003.

[9] MICHAEL J. BEST and JAROSLAVA HLOUSKOVA. *Quadratic Programming with Transaction Costs.* to appear in Computers and Operations Research.

[10] MICHAEL J. BEST and MARINA POTAPTCHIK. *Portfolio Optimization Using Nonsmooth Convex Transaction Costs.* Combinatorics and Optimization Research report 2004-30, December 2004.

[11] C. de BOOR. A Practical Guide to Splines. Springer, 2001.

[12] PHELIM P. BOYLE and XIAODONG LIN. *Optimal Portfolio Selection with Transaction Costs.* North American Actuarial Journal, vol.1, n.2, pp.27-39, 1997.

[13] BURKE J.V., LEWIS A.S. and OVERTON M.L. *Approximating Subdifferentials by Random Sampling of Gradients.* Mathematics of Operations Research 27, 567-584,(2002).

[14] E. W. CHENEY, A. A. GOLDSTEIN. *Newton's Method for Convex Programming and Tchebishef approximation.* Numerische Mathematik, vol. 1, pp. 253-268, 1959.

[15] CLARKE F.H. (1983). *Optimization and Nonsmooth Analysis.* Wiley.f18

[16] A.R. CONN, N. GOULD, M. LESCRENIER, and P.L. TOINT. Performance of a multifrontal scheme for partially separable optimization. In *Advances in optimization and numerical analysis (Oaxaca, 1992)*, volume 275 of *Math. Appl.*, pages 79–96. Kluwer Acad. Publ., Dordrecht, 1994.

[17] A.R. CONN, N. GOULD, and P.L. TOINT. Improving the decomposition of partially separable functions in the context of large-scale optimization: a first approach. In *Large scale optimization (Gainesville, FL, 1993)*, pages 82–94. Kluwer Acad. Publ., Dordrecht, 1994.

[18] G.B. DANTZIG, J. FOLKMAN and N. SHAPIRO. On the continuity of the minimum set of a continuous function. *Journal of Mathematical Analysis and its Applications,* 17: 519–548, 1967.

[19] M.J. DAYDÉ, J.Y. L'EXCELLENT, and N.I.M. GOULD. Element-by-element preconditioners for large partially separable optimization problems. *SIAM J. Sci. Comput.*, 18(6):1767–1787, 1997.

[20] Joydeep DUTTA. *Generalized Derivatives and Nonsmooth Optimization, a Finite Dimensional Tour.* Sociedad de Estadstica Investigacion Operativa, Top (2005) Vol. 13, No. 2, pp. 185-314.

[21] J. ELZINGA and T. G. MOORE. *A central cutting plane algorithm for the convex programming problem.* Mathematical Programming, vol.8, n.1, pp.134-145, 1975.

[22] E. ERDOĞAN, D. GOLDFARB, G. IYENGAR, Robust portfolio management. *CORC Tech. Report* TR-2004-11, IEOR Dept., Columbia Univ., NY, NY, USA, 2004.

[23] A.V. FIACCO. Introduction to Sensitivity and Stability Analysis in Nonlinear Programming. Academic Press, New York, 1983.

[24] A.V.FIACCO and G.P.MCCORMICK. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley and Sons, New York, Philadelphia 1990.

[25] D. GOLDFARB, G IYENGAR, Robust portfolio selection problems. *Mathematics of Operations Research*, **28** (2003)1–38.

[26] J.-L.GOFFIN, J.-P. VIAL, *Convex Nondifferentiable Optimization: A Survey Focused on the Analytic Center Cutting Plane Method*, Optimization Methods and Software, vol.17, n.5, 2002.

[27] A. GRIEWANK and P.L. TOINT. Numerical experiments with partially separable optimization problems. In *Numerical analysis (Dundee, 1983)*, volume 1066 of *Lecture Notes in Math.*, pages 203–220. Springer, Berlin, 1984.

[28] A.O. GRIEWANK and PH.L. TOINT. On the unconstrained optimization of partially separable functions. In M.J.D. Powell, editor, *Nonlinear Optimization.* Academic Press, London, 1982.

[29] J.-B. HIRIART-URRUTY, C. LEMARÉCHAL. *Fundamentals of Convex Analysis.* Springer, 2001.

[30] J. I.INGERSOLL. *Theory of financial decision making.* Rowman & Littlefield, 1987.

[31] J. E. KELLEY. *The Cutting Plane Method for Solving Convex Programs.* Journal of the SIAM, vol.8, pp.703-712, 1960.

[32] K. C. KIWIEL. *Methods of Descent for Nondifferentiable optimization.* Lecture Note in Mathematics, Springer-Verlag, 1985.

[33] K. C. KIWIEL. *A dual method for certain positive semidefinite quadratic programming problems.* SIAM Journal of Scientific and Statistical Computing, vol. 10, no. 1, pp.175–186, 1989.

[34] J. KREIMER and R. Y. RUBINSTEIN. Nondifferentiable optimization via smooth approximation: General analytical approach. *Annals of Operations Research* 39 (1992) 97–119.

[35] C. LEMARÉCHAL. *Nondifferentiable Optimization.* Handbooks In Operations Research And Management Science, Vol. 1, Elsevier Science Publishers B.V. 1989.

[36] C. LEMARÉCHA, J.-J. STRODIOT, and A. BIHAIN, *On a bundle algorithm for nonsmooth optimization.* Nonlinear Programming 4, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds., Academic Press, New York, 1991.

[37] A. LI. Some Applications of Symmetric Cone Programming in Financial Mathematics. M.Math. Essay, Dept. of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, Ontario, Canada, April 2005.
http://www.math.uwaterloo.ca/CandO_Dept/program_of_studies/graduate/MMathEssays.sht

[38] M. S. LOBO. *Convex and Robust Optimization with Applications in Finance.* PhD. thesis, Stanford University, USA, 2000.

[39] M.S. LOBO, M. FAZEL and S. BOYD. Portfolio optimization with linear and fixed transaction costs. *Annals of Operations Research* to appear, 2006.

[40] M. M. MAKELA. *Survey on Bundle Methods for Nonsmooth Optimization.* Optimization Methods and Software, Vol.17(1), pp1-29, 2001.

[41] H. MARKOWITZ. *Portfolio selection.* Journal of Finance, vol. 7, pp.77-91, 1952.

[42] K. G. MURTY. Linear Complementarity, Linear and Nonlinear Programming. Heldermann Verlag, Berlin, 1988.

[43] Yu. E. NESTEROV and A. S. NEMIROVSKII. *Interior-Point Polynomial Algorithms in Convex programming.* SIAM, Philadelphia, PA, USA, 1994.

[44] Yu. E. NESTEROV and M. J. TODD. *Self-scaled Barriers and Interior-Point Methods for Convex Programming.* Mathematics of Operations Research, vol. 22, pp.1-46, 1997.

[45] C. VAN DE PANNE and A. WHINSON. *The symmetric formulation of the simplex method for quadratic programming.* Econometrica vol.37, pp507-527, 1969.

[46] A.F. PEROLD. Large-scale portfolio optimization. *Management Sci.*, 30(10):1143–1160, 1984.

[47] H. QI and X. YANG. Regularity and well-posedness of a dual program for convex best C^1-spline interpolation. Report, University of Southhampton, Southhampton, England, 2004.

[48] R. TYRRELL ROCKAFELLAR. *Convex Analysis* Princeton University Press, Princeton, 1970.

[49] ROCKAFELLAR, R. T. and SUN, J. *A finite simplex-active-set method for monotropic piecewise quadratic programming.* Advances in optimization and

approximation, Nonconvex Optimization and it's Applications, 1, Kluwer Academic Publishers, Dordrecht, pp.275-292, 1994.

[50] N. Z. SHOR. *Minimization Methods for Nondifferentiable Functions.* Springer-Verlag, Berlin, 1985.

[51] N. Z. SHOR. *Nondifferentiable Optimization and Polynomial Problems.* Kluwer, Dordrecht, the Netherlands, 1998.

[52] J.B. SCHATTMAN. Portfolio selection under nonconvex transaction costs and capital gains taxes. PhD. thesis, Rutgers Center for Operations Research, Rutgers University, USA, 2000.

[53] J.W. SCHMIDT. Dual algorithms for solving convex partially separable optimization problems. *Jahresber. Deutsch. Math.-Verein.*, 94(1):40–62, 1992.

[54] P.L. TOINT. Global convergence of the partitioned BFGS algorithm for convex partially separable optimization. *Math. Programming*, 36(3):290–306, 1986.

[55] ROBERT J. VANDERBEI, *Linear Programming: Foundations and Extensions.* Kluwer Academic Publishers, Boston, MA, 2001.

[56] H. WOLKOWICZ, R. SAIGAL and L. VANDENBERGHE, editors. *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications,.* Kluwer Academic Publishers, Boston, MA, 2000.