RWTH AACHEN UNIVERSITY

MASTERS THESIS

Fast Projection on Birkhoff polytope

Author:
Mohammad SAIFULLAH

Supervisor:
Dr. Rolf BARDELI

A thesis submitted in fulfilment of the requirements
for the degree of Masters of Science

in the

Research Group Name
Fraunhofer IAIS

February 23, 2016

Declaration of Authorship

I, Mohammad SAIFULLAH, declare that this thesis titled, "Fast Projection on Birkhoff polytope" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.

- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.

- Where I have consulted the published work of others, this is always clearly attributed.

- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.

- I have acknowledged all main sources of help.

- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

Abstract

Masters of Science

Fast Projection on Birkhoff polytope

by Mohammad SAIFULLAH

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too. . .

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

List of Abbreviations

LAH List Abbreviations Here
WSF What (it) Stands For

Physical Constants

Speed of Light $c = 2.99792458 \times 10^8 \, \mathrm{m \, s^{-1}}$ (exact)

List of Symbols

a	distance	m
P	power	W $(\mathrm{J\,s^{-1}})$
ω	angular frequency	rad

For/Dedicated to/To my…

Chapter 1

Introduction

Let us imagine a science fiction. Astronauts are coming back from World 2.0 after successfully completing their mission. All the crew were so happy that they did not notice a large Asteroid is coming directly towards their space shuttle. When they have detected the astroid, it is already too late to avoid it's axis. So the only possibility is to use their advanced weapon system to destroy the Asteroid. But the battery requires considerable amount of time to initiate the weapon. Luckily they have alternative solar cells and they are equally distributed over the spaceship. Astronauts quickly analyze the situation. They found that, they are crossing a bright star at the moment and they have only chance if they rotate the shuttle such that it maximizes the facing area with respect to the light source. Now, Everything goes accordingly and the space shuttle could avoid the collision. A new problem appears when astonauts identify toxic radiations from the nearby star. So they need to control their appearance to the star such that it can minimise the rays directly falling onto the surface.

This is the problem where a source in $\mathbb{R}^3$ can be controlled in such a way that, the shadow area to a particular direction of projection can be minimized or maximized. Scientifically the source is considered inert and objective is to identify the direction that optimizes the area of the sources's projection on to another object orthogonal to the direction.

Although Projection to a subspace is mathematically expressed as matrix operation, it is not always obvious how to find optimal projection to a subspace for a specific application. This kind of problem appears in for example analysis of Astronomical, linguistic data, also while designing manifold algorithms. The goal of this thesis is to find a fast optimised projection to special polytope, named birkhoff polytope.

Chapter 2

The Birkhoff Polytope

In this Chapter I am going to describe about Birkhoff polytope B_n which is sometimes considered to be one of the most important polytopes in many sphere. Before that I will define some basic types.

2.1 Hypermatrix

A generalisation of matrix to an $n_1 \times n_1 \times ...$ array of numbers.
Example: 3-dimensional hypermatrix(cube). It is a $2 \times 2 \times 2$ matrix over $\mathbb{R}$.

FIGURE 2.1: 3-dimensional hypermatrix(cube).

2.2 Doubly-Stochastic Matrix

Doubly Stochastic Matrix is a square matrix of non-negative real numbers, each of whose rows and columns sum to 1. If rows and columns of some matrices sum to n, it can also be considered as doubly stochastic matrix because we can just device all the entries by n, which results to true Doubly Stochastic.
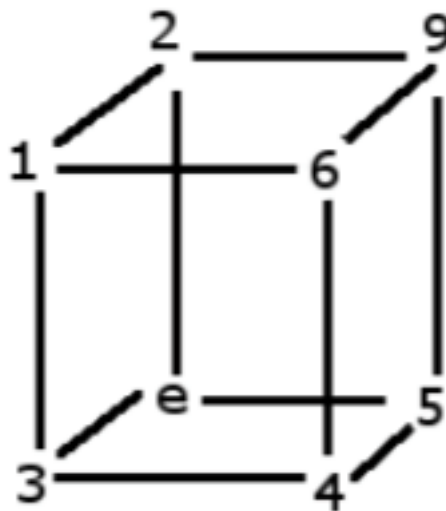
Examples:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1/6 & 5/6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1/6 & 0 & 5/6 \\ 5/6 & 0 & 0 & 1/6 \end{bmatrix}, \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

2.3 Permutation Matrix

A matrix obtained by permuting the rows of an $n \times n$ identity matrix according to some permutation of the numbers 1 to n. So the number of $n \times n$ permutation matrics is $n!$.

The permutation matrics of order 3 are:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

2.4 Polytopes

In elementary geometry, a polytope is a geometric object with flat sides, and may exist in any general number of dimensions n as an n-dimensional polytope or n-polytope. For example a two-dimensional polygon is a 2-polytope and a three-dimensional polyhedron is a 3-polytope

Mathematically, A polytope $P \subseteq \mathbb{R}^d$ is the convex hull $P = conv(v_1, ..., v_k)$ of a finite set of points $v_1, ..., v_k \in \mathbb{R}^d$. Dually any polytope can be written as the bounded intersection of a finite number of affine half-spaces in the form $P = \{x \mid Ax \le b\}$

2.4.1 Elements of polytope

A proper face of F of a polytope P is the intersection of P with and affine hyperplane H such that P is completely contained in one of the closed half spaces defined by H. The empty set and the polytope P a face of P. Any face F is itself polytope. The dimension of a polytope $P \subseteq \mathbb{R}^d$ is the dimension of the minimum affine space containin it. It is full dimensional if its dimension is d.

0-dimensional faces of P are called *vertices*, 1-dimensional faces are edges. Proper faces of maximal dimensional are called facets. P is the convex hull of its vertices, and the vertices of any face are subset of the vertices of P. Thus, a polytope has only a finite number of faces. Let f_i be the number of i-dimensional faces of $P, 0 \le i \le dimP - 1$. The f-vector of a d-dimensional polytope P is the non-negative integral vector $f(P) = (f_0, ..., f_(d-1)$.

The face lattice or combinatorial type $\mathcal{L}(P)$ of a polytope P is the partially ordered set of all faces of P (including the empty face and P itself). This defines Eulerian lattice. Figure shows 2.1 this lattice.

It contains all combinatorial information of the polytope. Two polytopes P,P' are combinatorially isomorphic or have the same combinatorial type if their face lattices are isomorphic as posets.

An r-dimensional simplex (or r-simplex) is the convex hull of r+1 affinely independent points in $\mathbb{R}^d$. A polytope is called simplicial if all facets are simplices. It is simple if the dual is simplicial. Equally, a d-dimensilanl polytope P is simple if each vertex is incident to precisely d edges. The d-dimensional 0/1-cube C^d is the convex hull of all

d-dimensinal 0/1-vectors. This is a simple d-polytope with 2^d vertices and 2d facets. More generally we denote by a d-cube any d-dimensional polytope that is combinatorially isomorphic to the 0/1-cube (it need not be full dimensional).

2.4.2 Properties of Polytope

Let $P_1 \subset \mathbb{R}^{d_1}$ and $P_2 \subset \mathbb{R}^{d_2}$ be two (geometrically realised) polytopes with vertex sets $V(P_1) = \{v_1, ..., v_k$ and $V(P_2) = \{w_1, ..., w_l\}$. With $0^{(d)}$ we denote he d-dimensional zero vector.

The (geometric) product of P_1 and P_2 is the polytope

$$P_1 \times P_2 = conv((v_i, w_i) \in \mathbb{R}^{d_1+d_2} \mid 1 \le i \le k, i \le j \le l) \tag{2.1}$$

This is the same as the set of all points (v, w) for $v \in P_1$ and $w \in P_2$. The (geometric) join of P_1 and P_2 is the polytope

$$P_1 \star P_2 := conv(P_1 \times \{0^{d_2}\} \times \{0\} \cup \{0^{d_1}\} \times P_2 \times \{1\}) \subseteq \mathbb{R}^{d_1+d_2+1} \tag{2.2}$$

More generally we say that a polytope P is a product or join of two polytopes P_1 and P_2, if P is combinatorially isomorphic to the geometric prouct or geometric join of some realisations of the face lattices P_1, or P_2.

If F is face of a polytope $P = \{x \mid Ax \le b\} \subseteq \mathbb{R}^d$ and $\langle c, x \rangle \le d$ a linear funtional defining F, then the $wedgewedge_F(P)$ of P over F is defined to be the polytope.

$$wedge_F(P) = \{(x, x_0) \in \mathbb{R}^{d+1} \mid Ax \le b, 0 \le x_0 \le d - \langle c, x \rangle\} \tag{2.3}$$

Again, we say more generally that P is wedge of a polytope Q over some face F of Q if P is combinatorially equivalently to $wedge_F(Q)$

2.5 Birkhoff Polytope

Birkhoff polytope B_n is sometimes considered to be one of the most important polytopes in many sphere. Birkhoff polytope is also called assignment polytope, the polytope of doubly stochastic matrices, or the perfect matching polytope of complete bipartite graph $K_{(n, n)}$,transportation polytope. It surprisingly appears in various branches of mathematics from geometry to enumerative combinatorics to optimisation theory to Statistics.

The Birkhoff polytope B_n is the convex hull of all $(n \times n)$ permutation metrices, i.e.matrices which consists precisely one 1 in every row and column, and zeros at all places. Equivalently, B_n is the set of all non-negative $(n \times n)$-metrices, whose rows and columns all sum to 1, or the perfect matching polytope of the complete bipartite graph $K_{(n, n)}$. The Birkhoff polytope B_n has dimension $(n - 1)^2$ with $n!$ vertices and n_2 facets. The Birkhoff-von Neumann Theoram illustrated, B_n can be understood as the intersection of the positive orthant with a family of of hyperplanes.

Birkhoff polytopes are widely studied as a class of polytopes in the area of optimisation, statistics, enumerative combinatorics or representation theory. Despite, Combinatorial and Geometrical structure of Birkhoff polytope and its algorithmic treatment are still open to disover.

A Birkhoff polytope B_n is a polytope defined by the following equations and inequalities:

$$a_{i,j} \geqslant 0, \sum_{i=1}^{n} a_{i,j} = 1, \sum_{j=1}^{n} a_{i,j} = 1 \text{ for all } 1 \leqslant i, j \leqslant n. \tag{2.4}$$

($a_{i,j}$ can be thought of as $n \times n$ doubly stochastic matrices. It can be realise that B_n has dimension $(n-1)^2$ as values of $a_{i,j}, 1 \leqslant i, j \leqslant n$ determine the rest. Different way helps to realise that vertices of P_n are permutation matrices.

2.6 Properties of B_n

Here I am going to describe some of the properties of Birkhoff polytope.

2.6.1 Vetices

The Birkhoff polytope has $n!$ vertices. This was derived from the Birkhoff-von Neumann theorem.

2.6.2 Edges

The edges of the Birkhoff polytope corresponds to pairs of permutations differing by a cycle:

(σ, ω) such that $\sigma^{-1}\omega$ is a cycle.

This implies that the graph of B_n is a Caley graph of the symmetric group S_n. This ialso implies that the graph of B_3 is a complete graph K_6, and thus B_3 is a neighbourly prototype.

2.6.3 Facets

The Birkhoff polytope lies within and $(n^2 - 2n + 1)$-dimensional affine subspace of the n^2-dimensional space of all $n \times n$ matrices: this subspace is determined by the linear equality constraints that the sum of each row and each column be one. Within this subspace, it is defined by n^2 linear inequalities, one for each coordinate of the matrix, specifying that the coordinate be non-negative. Therefore, it has exactly n^2 facets.

2.6.4 Symmetries

The Birkhoff polytop B_n is both vertex-transitive and facet-transitive. This is not regular for $n > 2$.

2.6.5 Volume

One of the hardest open problem is to find the volume of a Birkhoff polytopes. Volume calculation was possible for $n \leqslant 10$. It is known to be equal to the volume of polytope associated with standard Young tabuleaux. The following asymptotic formula was founded by Rodney Canifeld and Brenden McKay:

$$vol(B_n) = \exp(-(n-1)^2 \ln n + n^2 - (n - \frac{1}{2}) \ln(2\pi) + \frac{1}{3} + \mathcal{O}(1)) \tag{2.5}$$

Chapter 3

Quadratic Programming

An optimisation problem with a quadratic objective function and linear constraints is called a quadratic program. Problem of this type are important in their own right, and they also arise as subproblems in methods for general constrained optimisations.

3.1 Definition of QP

The general quadratic program(QP) can be stated as

$$
\begin{aligned}
\min_{x} \quad & q(x) = \frac{1}{2}x^T G x + x^T c \\
\text{subject to} \quad & a_i^T x = b_i && i \in \mathbb{E} \\
& a_i^T x \geqslant b_i && i \in \mathbb{I}
\end{aligned}
\tag{3.1}
$$

where G is a symmetric $n \times n$ matrix, E and I are finite sets of indices, and c,x, and $a_i, i \in E \cup I$, are vectors in $\mathbb{R}^n$. Quadratic programs can always be solved (or shown to be infeasible) in a finite amount of computation, but the effort required to find a solution depends strongly on the characteristics of the objective function and the number of inequality constraints. If the Hessian matrix G is positive semidefinite, we say that 3.1 is a convex QP, and in this case the problem is often similar in difficulty to a linear program. (Strictly convex QPs are those in which G is positive definite.) Nonconvex QPs, in which G is an indefinite matrix, can be more challenging because they can have several stationary points and local minima.

In this chapter we will try to show different kinds of quadratic programs but we will mainly focus on convex quadratic program and different algorithms of convex quadratic programs.

3.2 Classification of QP's

Some classification of quadratic program's:

- Unconstrianed QP

- Box constrained QP

- Equality constrained QP

- Inequality constrained QP.

Important Definitions

Logarithmic Barrier

Consider inequalities $Ax \leqslant b$ with A of size $m \times n$ and with rows a_i^T. Define

$$P = \{x \mid Ax \leqslant b\} \text{ and } P^0 = \{x \mid Ax < b\}$$

logarithmic barrier for the inequalities $Ax \leqslant b$:

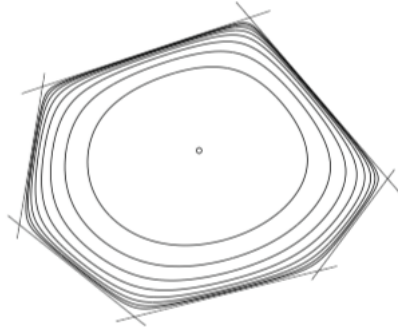$$\phi(x) = -\sum_{i=1}^{m} \log(b_i - a_i^T x) \quad \text{with domain } P^0$$

FIGURE 3.1: Logarithmic barrier function.

Gradient

Gradient $\nabla\phi(x)$ is the n-vector with $\nabla\phi(x)_i = \frac{\delta\phi(x)}{\delta x_i}$

$$\nabla\phi(x) = \sum_{k=1}^{m} \frac{1}{b_k - a_k^T x} a_k = A^T d_x$$

d_x denotes the positive $m-$vector

$$d_x = \left(\frac{1}{b_1 - a_1^T x}, ..., \frac{1}{b_m - a_m^T x} \right)$$

KKT System

KKT System is

Hessian Matrix

Hessian Matrix Gradient $\nabla^2\phi(x)$ is the $n \times n$-matrix with $\nabla\phi(x)_{ij} = \frac{\delta^2\phi(x)}{\delta x_i \delta x_j}$

$$\nabla^2\phi(x) = \sum_{k=1}^{m} \frac{1}{(b_k - a_k^T x)^2} a_k a_k^T = A^T diag(d_x)^2 A$$

Central Path

Consider the linear programming problem in standard form:

$$\min c^T x, \quad \text{subject to} Ax = b, \geqslant 0$$

where c and x are vectors in $\mathbb{R}^n$, b is a vector in $\mathbb{R}^m$, and A is an $m \times n$ matrix with full row rank. The dual problem of the above problem is

$$\max b^T \lambda, \quad \text{subject to} A^T \lambda + s = c, x \geqslant 0$$

where λ is a vector in $\mathbb{R}^m$ and s is a vector in $\mathbb{R}^n$ The primal-dual feasible set $\mathcal{F}$ and strictly feasible set $\mathcal{F}^0$ are defined as

$$\mathcal{F} = \{(x, \lambda, s) \mid Ax = b, A^T \lambda + s = c, (x, s) \geqslant 0\}$$
$$\mathcal{F}^0 = \{(x, \lambda, s) \mid Ax = b, A^T \lambda + s = c, (x, s) > 0\}$$

The central path C is an arc of strictly feasible points. It is parameterized by a scalar $\eta > 0$, and each point $(x_\tau, \lambda_\tau, s_\tau) \in \mathcal{C}$ satisfies the following equations:

$$\begin{aligned} A^T \lambda + s &= c, \\ Ax = b, x_i, s_i &= \tau \quad i = 1, 2, ...n. \\ (x, s) &> 0. \end{aligned}$$

So the central path is defined as:

$$\mathcal{C} = \{(x_\tau, \lambda_\tau, s_\tau) \mid \tau > 0\}$$

Another way of defining $\mathcal{C}$ is:

$$F(x_\tau, \lambda_\tau, s_\tau) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}, (x_\tau, s_\tau) > 0$$

the conditions approximate more and more colsely as τ goes to zero. If $\mathcal{C}$ converges to anything as $\tau \downarrow 0$, it must converge to a primal-dual solution of the linear program.

Active-set

The Active set $\mathbb{A}(x)$ at any feasible x consists of the equality constraint indices from ε together with the indices of the inequality constraints i for which $c_i(x) = 0$; that is,

$$\mathbb{A}(x) = \varepsilon \cup i \in \mathbb{I} \mid c_i(x) = 0$$

At a feasible point x, the inequality constraint $i \in \mathbb{I}$ is said to be active if $c_i(x) = 0$ and inactive if the strict inequality $c_i(x) > 0$ is satisfied.

3.3 Equality-Constrained Quadratic Programs

We start this section with of algorithms for quadratic programming by considering the case of equality constrained

Properties of Equality-constrained QPs

To make it simple, we write the equality constraint in a form of matrix and define it as follows:

$$\min_{x} \quad q(x) = \frac{1}{2}x^T G x + x^T c \tag{3.2}$$
$$\text{subject to} \quad Ax = b$$

where A is the $m \times n$ Jacobian of constraints (with $m \leqslant n$) whose rows are $a_i^T, i \in \mathbb{E}$ and b is the vecotor in $\mathbb{R}^n$ whose componants are $b_i, i \in \mathbb{E}$. Currently, we consider that A has a full row rank (rank m) so that the constraints are consistent.

The first-order necessary conditions for x^* to be a solution of 3.2 state that there is a vector λ^* such that the following system of equations is satisfied:

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda* \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix} \tag{3.3}$$

These conditions are a consequence of the general result for first-order optimality conditions. λ^* is called the vector of Lagrange multipliers. In 3.3 we can write x^* as $x^* = x + p$ which makes it useful for computation, where x is some estimate of the solution and p is the desired step. By introducing this and rearranging the equations, we obtain

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda* \end{bmatrix} = \begin{bmatrix} g \\ h \end{bmatrix} \tag{3.4}$$

where

$$h = Ax - b, g = c + Gx, \quad p = x^* - x. \tag{3.5}$$

The matrix 3.4 is called the Karush-Kuhn-Tucker (KKT) matrix, and the following result gives conditions under which it is nonsingular. We will use Z to denote the $n \times (n-m)$ matrix whose columns are basis for the null space of A. That is, z has full rank and satisfies $AZ = 0$.

> **Lemma**
>
> **Lemma 3.3.1.** *Let A have full row rank, and assume that the reduce Hessian matrix $Z^T G Z$ is positive definite. Then the KKT matrix*
>
> $$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \tag{3.6}$$
>
> *is nonsingular, and hence there is a unique vector pair (x^*, λ^*) satisfying 3.3.*

So, when the conditions of the 3.3.1 is satisfied, there is a unique vector pair (x^*, λ^*) that satiesfies the first-order necessary condition for 3.2. In fact, the second order sufficient conditions are also satisfied at (x^*, λ^*), so x^* is a strict local minimizer of 3.2. In fact we can use a direct argument to show that x^* is a global solution of 3.2.

> **Theorem**
>
> **Theorem 3.3.2.** *Let A have full row rank and assume that the reduced-Hessian matrix $Z^T G Z$ is positive definite. Then the vector x^* satisfying 3.3 is the unique global solution of 3.2.*

Proof. Let x be any other feasible point (satisfying $Ax = b$), and as before, let p denote the difference $x^* - x$. Since $Ax^* = Ax = b$, we have that $Ap = 0$. By substituting into the objective function 3.2, we get

$$
\begin{aligned}
q(x) &= \frac{1}{2}(x^* - p)^T G(x^* - p) + C^T(x^* - p) \\
&= \frac{1}{2}p^T Gp - p^T Gx^* - C^T p + q(x^*)
\end{aligned}
\tag{3.7}
$$

From 3.3 we have that $Gx^* = -c + A^T\lambda^*$, so from $Ap = 0$ we have that

$$
p^T Gx^* = p^T(-c) + A^T\lambda^* = -p^T c.
\tag{3.8}
$$

By substituting this relation into 3.7, we obtain

$$
q(x) = \frac{1}{2}p^T Gp + q(x^*).
\tag{3.9}
$$

Since p lies in the null space of A, we can write $p = Zu$ for some vector $u \in \mathbb{R}^{n-m}$, so that

$$
q(x) = \frac{1}{2}u^T Z^T GZu + q(x^*).
\tag{3.10}
$$

By positive definiteness of $Z^T GZ$, we conclude that $q(x) > q(x^*)$ except when $u = 0$, that is, when $x = x*$. Therefore, x^* is the unique global solution of 3.2 □

When the reduced Hessian matrix $Z^T GZ$ is positive semidefinite with zero eigenvalues, the vector x^* satisfying 3.4 is a local minimizer but not a strict local minimizer. If the reduced Hessian has negative eigenvalues, then x^* is only a stationary point, not a local minimizer.

3.4 Direct Solution of the KKT System

In this section we discuss the efficient methods of solving KKT system. The KKT system is always indefinite if $m \geqslant 1$. We can apply direct techniques to solve indefinite KKT system.

Factoring the Full Scale System

One option for solving KKT system is to perform triangular factorization on the full KKT matrix and then perform backward and forward substitution with the triangular factors. It is not possible to apply Cholesky factorization as it is indefinite.Another option can be Gaussian Elimination to obtain the L and U factors, but this method does not consider the symmetry.

So, the most effective strategy is to use symmetric indefinite factorization which has the form of:

$$
P^T KP = LDL^T
\tag{3.11}
$$

where P is an appropriately chosen permutation matrix. L is lower triangular with $diag(L) = I$ and D is block diagonal. Based on 3.11, the KKT system 3.4 is solved as

follows:

$$
\begin{aligned}
solve \quad & Ly = P^T \begin{bmatrix} g \\ h \end{bmatrix} \\
solve \quad & D\hat{y} = y \\
solve \quad & L^T \bar{y} = \hat{y} \\
set \quad & \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = P\bar{y}
\end{aligned}
\tag{3.12}
$$

This approach of factoring the full $(n + m) \times (n + m)$ KKt matrix is quite effective on many problems. It can be expensive when the permutation matrix P are not able to maintain sparsity in the L factor.

Range-space Approach

The range-space approach is useful when $G \in \mathbb{R}^{n \times n}$ is symmetric positive definite. We can multiply the first part of the equation 3.4 by AG^{-1} and then subtract the second part to obtain a linear system in the vector λ^* alone:

$$
(AG^{-1}A^T \lambda^*) = (AG^{-1}g - h)
\tag{3.13}
$$

We solve this symmetric semidifinite system for λ^* and then recover p from the first equation in 3.4 by solving

$$
Gp = A^T \lambda^* - g
\tag{3.14}
$$

This approach requires us to perform operation with G^{-1}, as well as to compute the factorization of the $m \times n$ matrix $AG^{-1}A^T$. That is why it is useful when:

- G is well conditioned and easily invertible (e.g., G is diagonal or block-diagonal),

- B^{-1} is known explicitely (e.g., by means of a quasi-Newton updating formula),

- the number m of equality constraints is small.

Null-space Approach

The null-space approach does not require regularity of G and thus has a wider range of applicability than the range-space approach.

We assume that $A \in \mathbb{R}^{m \times n}$ has full row rank m and that $Z^T G Z$ is positive definite, where $Z \in \mathbb{R}^{n \times (n-m)}$ is the matrix whose columns span Ker A which can be computed by QR factorization.

We partition the vector x^* according to

$$
x^* = Y w_y + Z w_z
\tag{3.15}
$$

where $Y \in \mathbf{R}^{n \times m}$ is such that $[Y \quad Z] \in \mathbb{R}^{n \times n}$ is nonsingular and $w_y \in \mathbb{R}^m, w_Z \in \mathbb{R}^{n-m}$.

Substituting 3.15 into the 3.4, we get

$$
\begin{aligned}
Ax^* = AY w_Y + AZ w_Z &= c \\
AZ &= 0
\end{aligned}
\tag{3.16}
$$

i.e., $Y w_Y$ is a particular solution of $Ax = c$.

Since $A \in \mathbb{R}^{m \times n}$ has rank m and $[Y \quad Z] \in \mathbb{R}^{n \times n}$ is nonsingular, the product matrix $[Y \quad Z] = [AY \quad 0] \in \mathbb{R}^{m \times m}$ is non singular. Hence, w_Y is well determined by 3.16.

On the otherhand, substituting 3.15 into the first equation of 3.4, we get

$$GYw_Y + GZw_Z + A^T\lambda^* = b. \tag{3.17}$$

Multiplying by Z^T and observing $Z^T A^T = (AZ)^T = 0$ yields

$$Z^T GZw_Z = Z^T b - Z^T GYw_Y. \tag{3.18}$$

The reduced KKT system 3.18 can be solved by a Cholesky factorization of the reduced Hessian $Z^T BZ \in \mathbb{R}^{(n-m)\times(n-m)}$. Once w_Y and w_Z have been computed as the solution of 3.17 and 3.17, x^* is obtained according to 3.15.

Finally, the Lagrange multiplier turns out to be the solution of the linear system arising from multiplication of the equation 3.15 by Y^T:

$$(AY)^T\lambda^* = Y^T b - Y^T Gx^* \tag{3.19}$$

3.5 Iterative solution of the KKT System

Direct solution of the KKT system can be sometimes expensive, the possible alternative is iterative method to solve KKT system. one of the famous iterative method is Conjugate Gradient method. Although it is not recommended for solving the full system using the CG mathod because it can be unstable on systems that are not positive definite. An iterative solver can be applied either to the entire KKT system or, as in the null-space and range-space approach, use the special structure of the KKT matrix.

3.6 Inequality-Constrained Problems

Inequality-constrained quadratic programs are QPs which consists of inequality constraints and may or may not consist equality constraints. Several classes of algorithms for solving convex quadratic programs containing inequality and equality constraints are available. Active-set method and Interior-point are most famous for their accuracy and capability to solve large problems.

3.6.1 Properties of Inequality-constrained Problems

Optimality condition for inequality-constrained problems

Lagrangian for the problem 3.1 is:

$$\mathcal{L}(x,\lambda) = \frac{1}{2}x^T Gx + x^T c - \sum_{i\in\mathcal{I}\cup\varepsilon}\lambda_i(a_i^T x - b_i) \tag{3.20}$$

The active set $\mathcal{A}(x^*)$ consists of indices of the constraints for which equality holds at x^*:

$$\mathcal{A}(x^*) = \{i \in \varepsilon \cup \mathcal{I} \mid a_i^T x^* = b_i\} \tag{3.21}$$

According to the KKT condition for this problem, it is found that any solution x^* for some Lagrange multipliers $\lambda_i^*, i \in \mathcal{A}(x^*)$ of 3.1 satisfies the following conditions:

$$Gx^* + c - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* a_i = 0$$

$$\begin{aligned} a_i^T x^* &= b_i \quad \text{for all } i \in \mathcal{A}(x^*), \\ a_i^T x^* &\geqslant b_i \quad \text{for all } i \in \mathcal{I}/\mathcal{A}(x^*), \\ \lambda_i^* &\geqslant 0, \quad \text{for all } i \in \mathcal{I} \cap \mathcal{A}(x^*), \end{aligned} \qquad (3.22)$$

Incase of convex QP, that is when G is positive semidefinite, 3.22 are sufficient for x^* to be a global solution.

> **Theorem**
>
> **Theorem 3.6.1.** *If x^* satisfies the conditions 3.22 for some $\lambda_i^*, i \in \mathcal{A}(x^*)$, and G is positive semidefinite, then x^* is a global solution of 3.1.* ♣

Degeneracy

Degeneracy initiates difficulties for some optimisation algorithm. It appears in the following situation:

- the active constraint gradients $a_i, i \in \mathcal{A}(x^*)$, are linearly dependent at the solution x^*, and/or

- there is some index $i \in \mathcal{A}(x^*)$ such that all Lagrange multipliers satifying KKT conditions have $\lambda_i^* = 0$
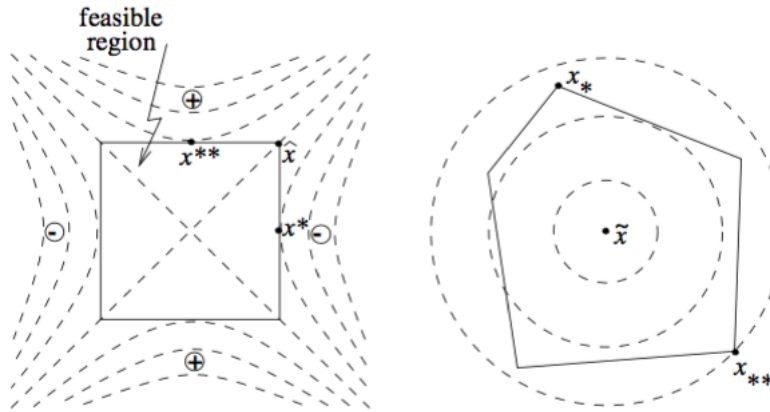

FIGURE 3.2: Nonconvex quadratic programs.

In the figure 3.3 two instances are visualized. There is only a single active constraint at the solution x^* in the left picture, that is lso an unconstrained minimizer of the objective function. According to KKT condition, $Gx^* + c = 0$, such that the lone Lagrange multiplier should be zero. 3 constraints are active at the solution x^* in the right-side image. As each of the three constraint gradients is a a vector in $\mathbb{R}^2$, they should be linearly independent.
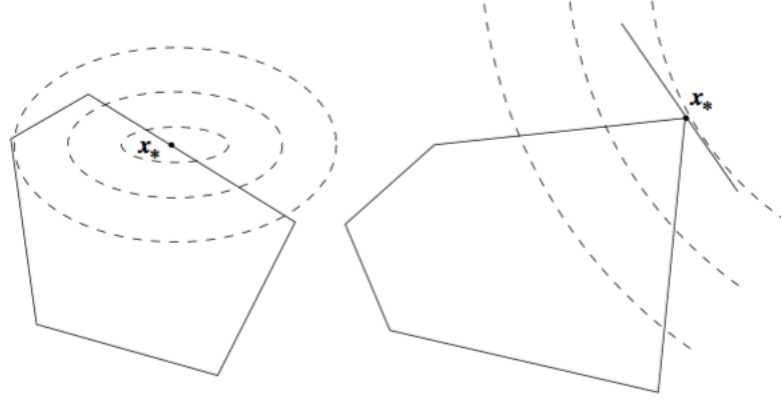
FIGURE 3.3: Degenerate solutions of quadratic programs.

Specifically for two reason Degeneracy can cause problems for optimization algorithms,

- First linear dependence of the active constraint gradients can cause numerical difficulties as several matrices which required to factor become rank deficient.

- Second when the problem contains weakly active constraints, it is difficult for the algorithm to determine whether these constraints are active at the solution.

3.7 Active-set Methods for Convex QPs

Active-set methods are important method for solving convex quadratic programs which consists equality and inequality constraints. Active set method starts by finding a feasible point during initial phase and then search for a solution along the edges and faces of the feasible set by solving a sequence of equality-constrained QPs.

If it was possible to know the contents of the active set earlier, it would be straight forward to get the solution by solving an equality-constrained QP of the form:

$$\min_{x} \quad q(x) = \frac{1}{2}x^T G x + x^T c$$
$$\text{subject to} \quad a_i^T x = b_i \qquad \forall i \in \mathcal{A}(x^*)$$

But usually it is not possible to know $\mathcal{A}(x^*)$ and termination of this set is a major challnage for algorithms.

Primal active-set method follows step from one iteration to another by solving a quadratic subproblem in which some of the equality constraints and all the inequality constraints are imposed as equalities which is referred to as the working set. Working set at kth iterate x_k is denoted by $\mathcal{W}_k$

Consider an iterate x_k and the working set $\mathcal{W}_k$, it is necessary to check if x_k minimizes the quadratic q in the subspace defined by working set. Otherwise step p is computed by solving an equality-constrained subproblem in which the constraints corresponding to the working set $\mathcal{W}_k$ are regarded as equalities and all other constraints are temporarily disregarded. We define this subproblem in terms of the step p:

$$p = x - x_k, \quad g_k = G x_k + c.$$

By substituting x with $(x_k + p)$ into 3.1,

$$q(x) = q(x_k + p) = \frac{1}{2}p^T G p + g_k^T p + \rho_k$$

ρ_k is independent of p. So without considering ρ_k, we can write the QP subproblem to be solved at kth iteration as:

$$\begin{aligned} \min_x \quad & \frac{1}{2}p^T G p + g_k^T p \\ \text{subject to} \quad & a_i^T p = 0 \qquad \forall i \in \mathcal{W}_k \end{aligned} \tag{3.23}$$

Solution of this subproblem is denoted by p_k. Consider optimal p_k is nonzero for the moment. We need to calculate displacement along this direction. If $x_k + p_k$ is feasible with respect to all the constraints, then we can set:

$$x_{k+1} = x_k + p_k$$

Otherwise, we set:

$$x_{k+1} = x_k + \alpha_k p_k \tag{3.24}$$

Where α_k is the step length and is chosen possible largest value in the range $[0, 1]$ for which all constraints are satisfied.

Selecting α_k

If $a_i^T p_k \geqslant 0$ for some $i \notin \mathcal{W}_k$, then for all $\alpha_k \geqslant 0$ we have $a_i^T(x_k + \alpha_k p_k) \geqslant a_i^T x_k \geqslant b_i$. So, constraint i will be satisfied for all nonnegative choices of the step-length parameter. Whenever $a_i^T p_k < 0$ for some $i \notin \mathcal{W}_k$, we have $a_i^T(x_k + \alpha_k p_k) \geqslant b_i$ only if

$$\alpha_k \leqslant \frac{b_i - a_i^T x_k}{a_i^T p_k}$$

So, to maximize the decrement of q, α_k should be as large as possible in $[0, 1]$ subject to retaining feasibility, so we get the following equation:

$$\alpha_k = \min \left(1, \min_{i \notin \mathcal{W}_k, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k} \right) \tag{3.25}$$

The constraints i for which the minimum is achieved called blocking constraints.

If $\alpha_k = 1$ and no new constraints are active at $x_k + \alpha_k p_k$, then there is no blocking constraints on this iteration.

If $\alpha_k < 1$, step along p_k was blocked by some constraints not in $\mathcal{W}_k$, $\mathcal{W}_{k+1}$ is built by adding one of the blocking constraints to $\mathcal{W}_k$

This method is repeated untill a point $\hat{x}$ has been achieved that minimizes the quadratic objective function over its current working set $\hat{\mathcal{W}}$. Identifying this point is not hard because the subproblem as solution $p = 0$. Since $p = 0$ satisfies the optimality condition 3.4 for 3.23, it is found that:

$$\sum_{i \in \hat{\mathcal{W}}} a_i \hat{\lambda}_i = g = G\hat{x} + c \tag{3.26}$$

for some Lagrange multipliers $\hat{\lambda}_i, i \in \hat{\mathcal{W}}$. It follows that x^* and λ^* satisfy the first KKT condition, if the multipliers are defined corresponding to the inequality constraints that are not in the working set to be zero. As there are some control imposed on the step length, x^* is also feasible with respect to all the constraints, so the second and third KKT conditions are satisfied at this point.

Considering the signs of the multipliers corresponding to the inequality constraints in the working set, that is, the indices $i \in \hat{\mathcal{W}} \cap \mathcal{I}$. The fourth KKT condition is also satisfied if these multipliers are all nonnegative. So it can be concluded that, $\hat{x}$ is a KKT point for the original problem 3.1. In fact, since G is positive semidefinite, we have from Theorem *** that $\hat{x}$ is a global solution of 3.1.

If on the other hand, if there exists some $j \in \hat{\mathcal{W}} \cap \mathcal{I}$, such that

$$\lambda^* < 0$$

That constraints has to be removed from the active set and solve new subproblem. This will decreases the objective function. The following theorem states that this strategy produces a direction p at the next iteration that is feasible with respect to the removed constraint.

> **Theorem**
>
> **Theorem 3.7.1.** *Suppose that the point $\hat{x}$ satisfies first-order conditions for the equality-constrained subproblem with working set $\hat{W}$; that is, equation 3.25 is satisfied along with $a_i^T \hat{x} = b_i$ for all $i \in \hat{W}$. Suppose, too, that the constraint gradients $a_i, i \in \hat{W}$, are linearly independent and that there is an index $j \in \hat{W}$ such that $\hat{lambda}_j < 0$.*
> *Let p be the solution obtained by dropping the constraint j and solving the following subproblem:*
>
> $$\begin{aligned} \min_p \quad & \frac{1}{2}p^T G p + (G\hat{x} + c)^T p, \\ \text{subject to} \quad & a_i^T p = 0 \qquad\qquad \forall i \in \hat{\mathcal{W}} \text{ with } i \neq j \end{aligned} \qquad (3.27)$$
>
> *Then p is a feasible direction for constraint j, that is, $a_j^T p \geqslant 0$. Moreover, if p satisfies second- order sufficient conditions for above equation, then we have that $a_j^T > 0$, and that p is a descent direction for $q(.)$.*

Whenever p_k obtained from 3.23 is nonzero and satisfies second-order sufficient optimality conditions for the current working set, it is a direction of strict descent for $q(.)$

> **Theorem**
>
> **Theorem 3.7.2.** *Suppose that the solution p_k of 3.23 is nonzero and satisfies the second order sufficient conditions for optimality for that problem. Then the function $q(.)$ is strictly decreasing along the direction p_k.*

So it can be concluded that, When G is positive definite-the second order sufficient conditions are satisfied for all feasible subproblems of the form 3.23. Hence, it follows from the result that a strict decrease in $q(.)$ can be obtained whenever $p_k \neq 0$.

Specification of the Active-set Method for Convex QP

The whole Active-set algorithm can be specified as following:

Algorithm 1 Active-set method for convex QP

1: **procedure** ACTIVESET
2:     Compute a feasible starting point x_0;
3:     Set $\mathcal{W}_0$ to be a subset of the active constraints at x_0;
4:     **for** $k = 0, 1, 2, ...$ **do**
5:         Solve 3.23 to find p_k;
6:         **if** $p_k == 0$ **then**
7:             Compute Lagrange multipliers $\hat{\lambda}$ that satisfy 3.23 with $\hat{\mathcal{W}} == \mathcal{W}_k$
8:             **if** $\hat{\lambda} \geqslant 0$ **then** for all $i \in \mathcal{W} \cap \mathcal{I}$
9:                 Stop with solution $x^* = x_k$
10:             **else**
11:                 $j \leftarrow argmin_{j \in \mathcal{W}_k \cap \mathcal{I}} \hat{\lambda}_j$
12:                 $x_{k+1} \leftarrow x_k$
13:                 $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \backslash \{j\}$
14:             **end if**
15:         **else**
16:             α_k from 3.25
17:             $x_{k+1} \leftarrow x_k + \alpha_k p_k$
18:             **if** there are blocking constraints **then**
19:                 obtain $\mathcal{W}_{k+1}$ by adding one of the blocking constraints to $\mathcal{W}_k$
20:             **else**
21:                 $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$
22:             **end if**
23:         **end if**
24:     **end for**
25: **end procedure**

To implement active-set method efficiently an important key is reuse of nformation from solving the equality-constrained subproblem at the next iteration. The only difference between two consecutive subproblems is that the working set grows or shrinks by a single component. Efficient codes perform updates of the matrix factorizations obtained at the previous iteration, rather than calculating them from scratch each time.

3.8   Interior-Point Method

Interior-point methods follow iterative approach which is a good candidate as alternative of active-set method. This method is also know as trajectory-following, path-following method. Here only convex quadratic programming problem with inequality constraints will be focused. It is easier to the problem as follows:

$$\min_x \quad q(x) = \frac{1}{2}x^T G x + x^T c$$
$$\text{subject to} \quad Ax \geqslant b \tag{3.28}$$

where $G \in \mathbb{R}^{n \times n}$ is symmetric, positive semidefinite, $A \in \mathbb{R}^{m \times n}$.

$$A = [a_i]_{i \in \mathcal{I}} \quad b = [b_i]_{i \in \mathcal{I}}, \quad \mathcal{I} = \{1, 2, 3..., m\}$$

KKT conditions for this notation can be written as follows:

$$
\begin{aligned}
Gx - A^T\lambda + c &= 0, \\
Ax - b &\geqslant 0, \\
(Ax - b)_i \lambda_i &= 0, \quad i = 1, 2, ..., m \\
\lambda &\geqslant 0.
\end{aligned}
\tag{3.29}
$$

By introducing the slack vector $y \geqslant 0$, conditions can be rewritten as:

$$
\begin{aligned}
Gx - A^T\lambda + c &= 0, \\
Ax - y - b &= 0, \\
y_i \lambda_i &= 0, \quad i = 1, 2, ..., m \\
(y, \lambda) &\geqslant 0,
\end{aligned}
\tag{3.30}
$$

As G is positive semidefinite, above KKT conditions are necessary and sufficient to solve convex quadratic program.

Let, current iterate (x, y, z) that satisfies $(y, \lambda) > 0$, a complementary measure μ can be defined as:

$$\mu = \frac{y^T\lambda}{m} \tag{3.31}$$

Derived path-following for the KKT conditions by considering the above KKT conditions:

$$F(x, y, \lambda : \sigma, \mu) = \begin{bmatrix} Gx - A^T\lambda + c \\ Ax - y - b \\ \mathcal{Y}\Lambda e - \sigma\mu e \end{bmatrix} = 0 \tag{3.32}$$

Where

$$\mathcal{Y} = diag(y_1, y_2, ..., y_m), \quad \Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_1), \quad e = (1, 1, ..., 1)^T$$

and $\sigma \in [0, 1]$. The solution of 3.27 for all positive values of σ and μ define the central path, which is a trajectory that leads to the solution to the QP as $\sigma\mu$ tends to zero.

After selecting μ and applying Newton's method to 3.27 the following linear system is achieved:

$$
\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & \mathcal{Y} \end{bmatrix}
\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} =
\begin{bmatrix} -r_d \\ -r_p \\ -\Lambda\mathcal{Y}e + \sigma\mu e \end{bmatrix}
\tag{3.33}
$$

where

$$r_d = Gx - A^T\lambda + c \quad r_p = Ax - y - b$$

The new iterate (x^+, y^+, λ^+) are obtained by means of

$$(x^+, y^+, \lambda^+) = (x, y, \lambda) + \alpha(\Delta x, \Delta y, \Delta\lambda)$$

α is chosen such that $(x^+, \lambda^+) > 0$ and possibly to satisfy various other conditions.

Chapter 4

Cholesky Factorization

Cholesky factorization is an important term in the feld of linear algebra. It is the process of decomposing a Hermitian, positive-definite matrix into product of a lower and upper triangular matrix and its conjygate transpose. Which is useful for numerical optimization and Monte Carlo simulation. It was named after the name the name of its discoverer Andre-Louis Cholesky. It is believed that cholesky decomposition is almost twice as efficient as the LU decomposition. for solving system of linear equation.

4.1 Definition and Existance

The Cholesky factorization is only defined for symmetric or Hermitian positive definite matrices.

Definition 4.1.1. A matrix $A \in \mathbb{R}^{m \times m}$ is symmetric positive definite (SPD) if and only if it is symmetric ($A^T = A$) and for all nonzero vectors $x \in \mathbb{R}^m$ it is the case that $x^T A x > 0$.

> **Theorem**
>
> **Theorem 4.1.1.** *Given a SPD matrix A there exist a lower triangular matrix L such that $A = LL^T$.*

The lower triangular matrix L is known as the Cholesky factor and LL^T is known as the Cholesky factorization of A. It is unique if the diagonal elements of L are restricted to be positive.

The converse holds trivially: if A can be written as LL^* for some invertible L, lower triangular or otherwise, then A is Hermitian and positive definite.

4.2 LDL Decomposition

Although the focus of this chapter is Cholesky factorization, it is worth to make an understanding about a closely related variant of the Cholesky factorization, the LDL decomposition.

It is represented as,

$$A = LDL^*$$

where L is a lower triangular matrix and D is a diagonal matrix.LDL decomposition relates to Cholesky decomposition by the following way

$$A = LDL^* = LD^{\frac{1}{2}}D^{\frac{1}{2}*}L^* = LD^{\frac{1}{2}}(LD^{\frac{1}{2}})^*$$

When LDL is efficiently implemented, it takes the same space and complexity to build and use. LDL method method is used for those cases where no Cholesky decomposition possible.

4.3   Application of Choleskey Factorization

The Cholesky decomposition is mostly used when numerical solution of linear equations $Ax = b$ is required. When A is symmetric and positive definite, it is possible to solve $Ax = b$ by first computing the Cholesky decomposition $A = LL^*$, then solving $Ly = b$ for y by forward substitution, and finally solving $L * x = y$ for x by back substitution.

The Cholesky decomposition helps to achieve superior efficiency and numerical stability. Compared to the LU decomposition, this can perform almost two times efficiently.

Linear Least Squares

Systems of the form $Ax = b$ with A symmetric and positive definite arise quite often in applications. For instance, the normal equations in linear least squares problems are of this form. It may also happen that matrix A comes from an energy functional which must be positive from physical considerations; this happens frequently in the numerical solution of partial differential equations.

Non-linear Optimization

Non-linear multi-variate functions may be minimized over their parameters using variants of Newton's method called quasi-Newton methods. At each iteration, the search takes a step s defined by solving $Hs = -g$ for s, where s is the step, g is the gradient vector of the function's partial first derivatives with respect to the parameters, and H is an approximation to the Hessian matrix of partial second derivatives formed by repeated rank 1 updates at each iteration. Two well-known update formulae are called Davidon–Fletcher–Powell (DFP) and Broyden–Fletcher–Goldfarb–Shanno (BFGS). Loss of the positive-definite condition through round-off error is avoided if rather than updating an approximation to the inverse of the Hessian, one updates the Cholesky decomposition of an approximation of the Hessian matrix itself.

Monte Carlo Simulation

The Cholesky decomposition is commonly used in the Monte Carlo method for simulating systems with multiple correlated variables: The correlation matrix is decomposed, to give the lower-triangular L. Applying this to a vector of uncorrelated samples, u, produces a sample vector Lu with the covariance properties of the system being modeled.

For a simplified example that shows the economy one gets from Cholesky's decomposition, say one needs to generate two correlated normal variables x_1 and x_2. All one needs to do is to generate two uncorrelated Gaussian random variables z_1 and z_2. We set $x_1 = z_1$ and $x_2 = \rho z_1 + \sqrt{1 - \rho^2} z_2$.

Matrix Inversion

The explicit inverse of a Hermitian matrix can be computed via Cholesky decomposition, in a manner similar to solving linear systems, using n^3 operations ($\frac{1}{2}n^3$ multiplications). The entire inversion can even be efficiently performed in-place.

A non-Hermitian matrix B can also be inverted using the following identity, where BB^* will always be Hermitian:

$$\mathbf{B}^{-1} = \mathbf{B}^*(\mathbf{BB}^*)^{-1}.$$

4.4 The Cholesky Algorithm

Most usual algorithm for Cholesky factorization $chol(A)$ can be derived as: (*Greek lowercase letters refers to scalars, lower case letter referes to vector and uppercase letters referes to matrices*). Partition

$$A = \left(\begin{array}{c|c} \alpha_{11} & \star \\ \hline a_{21} & A_{22} \end{array}\right) \quad \text{and } L = \left(\begin{array}{c|c} \lambda_{11} & 0 \\ \hline \lambda_{21} & L_{22} \end{array}\right).$$

By substituting these partitioned matrices into $A = LL^T$, it is found that:

$$\left(\begin{array}{c|c} \alpha_{11} & \star \\ \hline a_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c} \lambda_{11} & 0 \\ \hline \lambda_{21} & L_{22} \end{array}\right)\left(\begin{array}{c|c} \lambda_{11} & 0 \\ \hline \lambda_{21} & L_{22} \end{array}\right)^T = \left(\begin{array}{c|c} \lambda_{11}^2 & \star \\ \hline \lambda_{11}l_{21} & l_{21}l_{21}^T + L_{22}L_{22}^T \end{array}\right)$$

so that

$$\left(\begin{array}{c|c} \alpha_{11} = \lambda_{11}^2 & \star \\ \hline a_{21} = \lambda_{11}l_{21} & A_{22} = l_{21}l_{21}^T + L_{22}L_{22}^T \end{array}\right)$$

and hence

$$\left(\begin{array}{c|c} \lambda_{11} = \sqrt{\alpha_{11}} & \star \\ \hline l_{21} = a_{21}/\lambda_{11} & L_{22} = chol(A_{22} - l_{21}l_{21}^T) \end{array}\right)$$

These equalities directs to the algorithm

Algorithm 2 Cholesky factorization

1: **procedure** CHOLESKY(A)
2:     Partition $A \leftarrow \left(\begin{array}{c|c} \alpha_{11} & \star \\ \hline a_{21} & A_{22} \end{array}\right)$
3:     Overwrite $\alpha_{11} := \lambda_{11} = \sqrt{\alpha_{11}}$
4:     Overwrite $a_{21} := l_21 = a_{21}/\lambda_1 1$.
5:     Overwrite $A_{22} := A_{22} - l_{21}l_{21}^T$ (updating lower triangular part of $A_2 2$)
6:     Continue with $A = A_{22}$
7: **end procedure**

Blocked Variant

The elements of the factorization can grow arbitrarily. So to achieve high performance, the computation is cast in terms of matrix-matrix multiplication by blocked algorithms. The blocked version of the Cholesky algorithm can be derived by partitioning

$$A = \left(\begin{array}{c|c} A_{11} & \star \\ \hline A_{21} & A_{22} \end{array} \right) \quad \text{and } L = \left(\begin{array}{c|c} L_{11} & 0 \\ \hline L_{21} & L_{22} \end{array} \right).$$

where A and L are $n_b \times n_b$ sized matrix(n_b block size). by substituting these into $A = LL^T$,

$$\left(\begin{array}{c|c} A_{11} & \star \\ \hline A_{21} & A_{22} \end{array} \right) = \left(\begin{array}{c|c} L_{11} & 0 \\ \hline L_{21} & L_{22} \end{array} \right) \left(\begin{array}{c|c} L_{11} & 0 \\ \hline L_{21} & L_{22} \end{array} \right)^T = \left(\begin{array}{c|c} L_{11}L_{11}^T & \star \\ \hline L_{21}L_{11}^T & L_{21}L_{21}^T + L_{22}L_{22}^T \end{array} \right)$$

And at the end

$$\left(\begin{array}{c|c} L_{11} = cholesky(A_{11}) & \star \\ \hline L_{21} = A_{21}L_{11}^{-T} & L_{22} = cholesky(A_{22} - L_{21}L_{21}^T) \end{array} \right)$$

So blocked variant algorithm can be described as:

Algorithm 3 Cholesky factorization

1: **procedure** CHOLESKYBLK(A)
2:     Partition $A \leftarrow \left(\begin{array}{c|c} A_{11} & \star \\ \hline A_{21} & A_{22} \end{array} \right)$
3:     Overwrite $A_{11} := L_{11} = CholeskyBLK(A_{11})$
4:     Overwrite $A_{21} := L_21 = A_{21}L_{11}^T$.
5:     Overwrite $A_{22} := A_{22} - L_{21}L_{21}^T$ (updating lower triangular part of A_22)
6:     Continue with $A = A_{22}$
7: **end procedure**

Updating the Decomposition

More often the task of updating Cholesky decomposition arises. More descriptively, at some point Cholesky decomposition $A = LL^*$ of some matrix A has been computed, then a change on matrix A has been made to produce matrix $\hat{A}$, and now it is required to compute the Cholesky decomposition of the new matrix: $\hat{A} = \hat{L}\hat{L}^*$. Now question arises if it is possible to reuse the Cholesky decomposition of A which already has been computed to compute the Cholesky decomposition of $\hat{A}$

Rank-one Update

The situation where the updated matrix $\hat{A}$ is produced from matrix A by $\hat{A} = A + xx^*$ is refers to rank-one update.

Cost

The cost of the cholesky factorization of $A \in \mathbb{R}^{m \times m}$ can be calculated from steps:

- $\alpha := \sqrt{\alpha_{11}}$ negligible when k is large.

- $a_21 := a_{21}/\alpha_{11}$ approx. $(m - k - 1)$ flops.

- $A_{22} := A_{22} - triangular(a_{21}a_{21}^T)$: approx. $(m - k - 1)^2$ flops

---

Algorithm 4 Rank-one Update

1: **procedure** CHOLESKYRANKUPDATE(L, x)
2:     $n \leftarrow length(x)$
3:    **for** $k = 1 : n$ **do**
4:        $r \leftarrow L(power(L(k, k), 2) + power(x(k), 2))$
5:        $c \leftarrow r/L(k, k)$
6:        $s \leftarrow x(k)/L(k, k)$
7:        $L(k, k) \leftarrow r$
8:        $L(K + 1 : n, k) \leftarrow (L(k + 1 : n, k) + s * x(k + 1 : n))/c$
9:        $x(k + 1 : n) \leftarrow c * x(k + 1 : n) - s * L(k + 1 : n, k)$
10:   **end for**
11: **end procedure**

So,the total cost in form of flops is:

$$
Cost_{Cholesky}(m) \approx \underbrace{\sum_{k=0}^{m-1} (m - k - 1)^2}_{\text{Due to update of } A_{22}} + \underbrace{\sum_{k=0}^{m-1} (m - k - 1)}_{\text{Due to update of} a_{21}}
$$

$$
= \sum_{j=0}^{m-1} j^2 + \sum_{j=0}^{m-1} j
$$

$$
\approx \frac{1}{3}m^3 + \frac{1}{2}m^2
$$

$$
\approx \frac{1}{3}m^3
$$

It can be realised that, most computation cost is in update of A_{22}

4.5 Sparse Cholesky Factorization Algorithm

CVXOPT extends the built-in Python objects with two matrix objects: a `matrix` object for dense

Chapter 5

CVXOPT

In this chapter we will introduce you with CVXOPT library. This library is has implemented a lot of important methods which can be used to solve different optimization problems. In the first a general description about this library has been given, then short a description about its modules. At the end algorithm behind the CVXOPT quadratic cone solver has been described.

5.1 A First View into CVXOPT Library

CVXOPT is an opensource library implemented using Python and C for convex optimization. This library can be installed as python package and can be used with interactive Python interpreter, integrate into software as Python extension module, or on the command line by executing python script. The goal of CVXOPT is to make software development straightforward where convex optimization is required by providing package using the power of python as Python's extensive standard library.

5.2 CVXOPT Module Structure

CVXOPT extends the default python matrix objects into `matrix` for dense matrices and `spmatrix` for sparse matrices. According to the CVXOPT manual, CVXOPT is organised into following different module:

`cvxopt.blas`

This is the interface to most of the double-precision real and complex Basic Linear Algebra Subprograms. Operations performed using BLAS routines can be implemented as a form arithmatic operation which helps to achieve great simplicity. The main two advantage of BLAS interface over other python blas packages is that:

- some functions are not just implementation of basic matrix arithmatic. For example, BLAS includes functions that efficiently exploit symmetry or triangular matrix structure.

- BLAS module maintains performance difference with other libraries which is significant for large matrices.

`cvxopt.lapack`

Interface to dense double-precission real and complex linear equation solvers and eigen-value routines. This module includes methods for solving dense sets of linear equations, for the corresponding matrix factorizations, for sloving least-squares and least-norm problems, for QR factorization, for symmetric eigenvalue problems, singular value decomposition and Schur factorization.

`cvxopt.fftw`

Interface to the discrete transform routines from FFTW and contains routines for discrete Fourier, cosine, and sine transforms.

`cvxopt.amd`

Interface to the approximate minimum degree ordering routine from AMD.

`cvxopt.umfpack`

This module includes four functions for solving sparse non-symmetric sets of linear equations.

`cvxopt.cholmod`

It is an interface to the Cholesky factorization routines of the CHOLMOD package. It includes functions for Cholesky factorization of sparse positive definite matrices.

`cvxopt.solvers`

Convex optimization routines and optional interfaces to solvers from GLPK, MOSEK, and DSDP5

`cvxopt.modeling`

Routines for specifying and solving linear programs and convex optimization problems with piecewise-linear cost and constraint functions

`cvxopt.printing`

Contains functions and parameters that control how matrices are formatted.

5.3   CVXOPT Cone Programming Interface

CVXOPT considers convex optimization problems as the following form:

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2}x^T P x + q^T x \\
\text{subject to} \quad & Gx \leqslant h \\
& Ax = b
\end{aligned}
$$

The linear inequality is a generalized inequality with respect to proper convex cone which may include componentwise vector inequalities, second-order cone inequalities, and linear matrix inequalities. The main solvers are `conelp` and `coneqp` which are used to optimize linear and quadratic cost functions and problems are required to be strictly primal and dual feasible. As the context of the thesis is Quadratic programming, quadratic programming method of `coneqp` interface is focused here.

5.3.1 Quadratic Cone Programs

CVXOPT Quadratic cone program routine solves a pair of primal and dual quadratic cone programs:

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2}x^T P x + q^T x \\
\text{subject to} \quad & Gx + s = h \\
& Ax = b \\
& s \geqslant 0
\end{aligned}
\tag{5.1}
$$

with P positive semidefinite. And corresponding dual problem is

$$
\begin{aligned}
\text{maximize} \quad & -(\frac{1}{2})(q + G^T z + A^T y)^T P^{\dagger}(q + G^T z + A^T y) - h^T z - b^T y \\
\text{subject to} \quad & q + G^T z + A^T y \in \ \text{range}(P) \\
& s \geqslant 0
\end{aligned}
\tag{5.2}
$$

x is the primal variable, s is the slack variable. y and z are dual variables. The inequalities are interpreted as $s \in C$, $z \in C$, were C is a cone defined as a cartesian product of a nonnegative orthant, a number of second-order cones, and a number f positive semidefinite cones:

$$
C = C_0 \times C_1 \times ... \times C_M \times C_{M+1} \times ... \times C_{M+N}
$$

with

$$
\begin{aligned}
C_0 &= \{u \in \mathbb{R}^l \mid u_k \geqslant 0, \ k = 1, ..., l\}, \\
C_{K+1} &= \{(u_0, u_1) \in \mathbb{R} \times \mathbb{R}^{\tau_k - 1} \mid u_0 \geqslant \|u_1\|_2\}, \quad k = 0, ..., M - 1 \\
C_{K+M+1} &= \{\text{vec}(u) \mid u \in S^{t_k}\}, \qquad\qquad\qquad\qquad k = 0, ..., N - 2.
\end{aligned}
$$

According to CVXOPT, $vec(u)$ denotes a symmetric matrix u stored as a vector in column major order.

The CVXOPT typical `coneqp` interface look like the following:

```
cvxopt.solvers.coneqp(P,q[,G,h[,dims[,A,b[,initvals[,kktsolver]]]]])
```

`P` is a square dense or sparse real matrix, `q` is a single-column dense real matrix. `h` and `b` are real single-column dense matrices. `G` and `A` are real dense or sparse matrices. The default value of `G`, `h`, `A` and `b` are metrices with zero rows, meaning that there are no inequalities or equality constraints. It is possible to provide custom solver to solve linear equation (KKT equations). `coneqp` returns a dictionary that which contains the result and accuracy of the solution.

5.3.2 CVXOPT quadratic cone solver algorithm

The algorithm implemented in the `coneqp` solver is primal-dual path-following method based on Nesterov-Todd scaling. `coneqp` solver is built using the following assumptions:

Rank assumptions

It is assumed that,

$$\mathbf{rank}(A) = p, \quad \mathbf{rank}(\begin{bmatrix} P & A^T & G^T \end{bmatrix}) = n$$

where p is the row dimension and n is the dimension of x. that means the matrix:

$$\begin{bmatrix} P & A^T & G^T \\ A & 0 & 0 \\ G & 0 & -Q \end{bmatrix}$$

is nonsingular for any positive definite Q.

    If $\mathbf{rank}(A) < p$ the either the equality constraints in the primal problem are inconsistent or some of the equalities are redundant. If $\mathbf{rank}(\begin{bmatrix} P & A^T & G^T \end{bmatrix}) < n$, then either the eqaulity constraints in the dual problem are inconsistent or some are redundant and the number of primal vriables can be reduced.

    The CVXOPT solver can not identify which rank assumptions does not hold, it only rise an exception if the rank conditions are not satisfied.

Logarithmic berrier function

CVXOPT uses the following barrier function for C:

$$\phi(u) = \sum_{k=1}^{K} \phi_k(u_k), \phi_k(u) = \begin{cases} -\sum_{j=1}^{p} \log u_j & C_k = R^p \\ -(1/2)\log(u_0^2 - u_1^T u_1) & C_k = \mathcal{Q}_p \\ -\log \det \mathbf{mat}(u) & C_k = \mathcal{S}_p \end{cases}$$

where $\phi(tu) = \phi(u) - m\log t$ for $t > 0$ where

$$m = m_1 + ... + m_k, m_k = \begin{cases} p & C_k = R^p \\ 1 & C_k = \mathcal{Q}_p \\ p & C_k = \mathcal{S}_p \end{cases}$$

m is refers to the degree of the cone C.

Central path

In CVXOPT the central path for 5.3.2 is defined by a family of points (s, x, y, z) that saistfy

$$\begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} + \begin{bmatrix} P & A^T & G^T \\ A & 0 & 0 \\ G & 0 & -Q \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -c \\ b \\ h \end{bmatrix}, \quad (s, z) \succ 0, \quad z = -\mu g(s) \qquad (5.3)$$

for some μ 0. CVXOPT Primal-dual algorithms are based on an equivalent definition of central where primal and dual variables appear symmetrically. The symmetric

parametrization is achieved by writing $z = -\mu g(s)$ as $s \circ z = \mu \mathbf{e}$ and $\mathbf{e}$ is the vector $\mathbf{e} = (\mathbf{e}_1, ..., \mathbf{e}_k)$,

$$\mathbf{e}_k = \begin{cases} (1, 1, ..., 1) & C_k = R^p \\ (1, 0, ..., 0) & C_k = \mathcal{Q}_p \\ \mathbf{vec}(I_p) & C_k = \mathcal{S}_p \end{cases}$$

Central path described finally as:

$$\begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} + \begin{bmatrix} P & A^T & G^T \\ A & 0 & 0 \\ G & 0 & -Q \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -c \\ b \\ h \end{bmatrix}, \quad (s, z) \succ 0, \quad s \circ z = \mu \mathbf{e} \qquad (5.4)$$

Nesterov-Todd Scaling

A primal-dual scaling W is a linear transformation

$$\bar{s} = W^{-T} s, \quad \bar{z} = W z$$

that leaves the cone and the central path invariant, *i.e.*,

$$s \succ 0 \Leftrightarrow \bar{s} \succ, \quad z \succ \Leftrightarrow \bar{z} \succ 0, \quad s \circ z = \mu \mathbf{e} \Leftrightarrow \bar{s} \circ \bar{z} = \mu \mathbf{e}$$

When W is a scaling, central path equations has been written equvalently as the following form

$$\begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} + \begin{bmatrix} P & A^T & G^T \\ A & 0 & 0 \\ G & 0 & -Q \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -c \\ b \\ h \end{bmatrix}, \quad (s, z) \succ 0, \quad (Wz) \circ (W^{-T}s) = \mu \mathbf{e} \qquad (5.5)$$

Path-following Algorithm for Cone QPs

The algorithm implemented in CVXOPT *coneqp* is based on linearizing the central path equation 5.5. which is achieved from equation 5.4 after applying a scaling with a matrix W.

CVXOPT defines the current iterate by $(\hat{s}, \hat{x}, \hat{y}, \hat{z})$. Initial iterate value is $((\hat{s}, \hat{x}, \hat{y}, \hat{z}) = (s_0, x_0, y_0, z_0))$ where $s_0 \succ 0, z_0 \succ 0$. Neterov-Todd scalling W is calculated at $\hat{s}, \hat{z}$ and the scaled variable $\lambda := W^{-T}\hat{s} = W\hat{z}$

The algorithm follows the following steps.

1. *Euclidian residuals,gap,and stopping criteria*.Compute

$$\begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \hat{s} \end{bmatrix} + \begin{bmatrix} P & A^\mathsf{T} & G^\mathsf{T} \\ A & 0 & 0 \\ G & 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} \begin{bmatrix} c \\ -b \\ -h \end{bmatrix} \qquad (5.6)$$

and

$$\hat{\mu} = \frac{\hat{s}^\mathsf{T} \hat{z}}{m} = \frac{\lambda^\mathsf{T}}{m}$$

Terminate if $(s, x, y, z) = (\hat{s}, \hat{x}, \hat{y}, \hat{z})$ satiesfies (within some threshold value) the optimality conditions

$$
\begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix} = \begin{bmatrix} P & A^{\mathsf{T}} & G^{\mathsf{T}} \\ A & 0 & 0 \\ G & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} c \\ b \\ h \end{bmatrix}, \quad (s, z) \succeq 0, \quad z^{\mathsf{T}} s = 0.
$$

2. *Affine Direction.* Solve the linear equations:

$$
\begin{bmatrix} 0 \\ 0 \\ \Delta s_a \end{bmatrix} + \begin{bmatrix} P & A^{\mathsf{T}} & G^{\mathsf{T}} \\ A & 0 & 0 \\ G & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x_a \\ \Delta y_a \\ \delta z_a \end{bmatrix} = - \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}
\tag{5.7}
$$

$$
\lambda \circ (W \Delta z_a + W^{-\mathsf{T}} \Delta s_a) = -\lambda \circ \lambda.
$$

3. *Step size and centering parameter.* Cpmpute

$$
\alpha = \sup\{\alpha \in [0, 1] \mid (\hat{s}, \hat{z}) + \alpha(\Delta s_a, \Delta z_a) \succeq 0\}
$$
$$
= \sup\{\alpha \in [0, 1] \mid (\lambda, \lambda) + \alpha(W^{-T} \Delta s_a, W \Delta z_a) \succeq 0\}
$$
$$
\rho = \frac{(\hat{s} + \alpha \Delta s_a)^T (\hat{z} + \alpha \Delta z_a)^T}{\hat{s}^T \hat{z}}
$$
$$
= 1 - \alpha + \alpha^2 \frac{(W^{-T} \Delta s_a)(W \Delta z_a)}{\lambda^T \lambda}
$$
$$
\sigma = \max\left\{0, \min\{1, \rho\}\right\}^3
$$

4. *Combined direction.* Solve the linear equation

$$
\begin{bmatrix} 0 \\ 0 \\ \Delta s_a \end{bmatrix} + \begin{bmatrix} P & A^{\mathsf{T}} & G^{\mathsf{T}} \\ A & 0 & 0 \\ G & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x_a \\ \Delta y_a \\ \delta z_a \end{bmatrix} = -(1 - \eta) \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}
\tag{5.8}
$$

$$
\lambda \circ (W \Delta z_a + W^{-\mathsf{T}} \Delta s_a) = -\lambda \circ \lambda - \gamma (W^{-\mathsf{T}} \Delta s_a) \circ (W \Delta z_a) + \sigma \hat{\mu} \mathbf{e}
$$

Common choice for η are $\eta = 0$ and $\eta = \sigma$. The parameter η is 1 or 0, depending on whether or not a Mehrota correction is used. The default value is $\gamma = 1$

5. *Update iterators and scalling matrices,*

$$
(\hat{s}, \hat{x}, \hat{y}, \hat{z}) := (\hat{s}, \hat{x}, \hat{y}, \hat{z}) + \alpha(\Delta s, \Delta x, \Delta y, \Delta z)
$$

Where

$$
\alpha = \sup\left\{\alpha \in [0, 1] \mid (\lambda, \lambda) + \frac{\alpha}{0.99}(W^{-\mathsf{T}} \Delta s_a, W \Delta z_a) \succeq 0\right\}
$$

Compute the scalling matrix W for $\hat{s}, \hat{z}$ and the second variable $\lambda := W^{-\mathsf{T}} \hat{s} = W \hat{z}$

Initialization of the Algorithm

If primal and dual starting points $\hat{x}, \hat{s}, \hat{y}, \hat{z}$ are not specified during the calling the method, they are initialized as by the algorithm by the following way.

$\boxed{\texttt{coneqp}}$ algorithm solves the linear equation:

$$
\begin{bmatrix} P & A^T & G^T \\ A & 0 & 0 \\ G & 0 & -I \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} -c \\ b \\ h \end{bmatrix}
\tag{5.9}
$$

and takes $\hat{x} = x, \hat{y} = y$. The equation 5.9 gives the optimality conditions for the pair of primal and dual problems

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}x^T P x + c^T x \parallel s \parallel_2^2 \\ \text{subject to} \quad & Gx + s = h \\ & Ax = b \end{aligned}$$

and

$$\begin{aligned} \text{maximize} \quad & -(\frac{1}{2})w^T P w - h^T z - b^T y - (1/2) \parallel z \parallel_2^2 \\ \text{subject to} \quad & Pw + G^T z + A^T y + c = 0 \end{aligned}$$

The initial value of $\hat{s}$ is computed from the residuals $h - Gx = -z$ as

$$\hat{s} = \begin{cases} -z & \alpha_p < 0 \\ -z + (1 + \alpha_p)\mathbf{e} & \text{otherwise} \end{cases}$$

where $\alpha_p = \inf\{\alpha \mid -z + \alpha \mathbf{e} \succeq 0\}$. The initial value of z is:

$$\hat{s} = \begin{cases} z & \alpha_d < 0 \\ -z + (1 + \alpha_d)\mathbf{e} & \text{otherwise} \end{cases}$$

where $\alpha_d = \inf\{\alpha \mid z + \alpha \mathbf{e} \succeq 0\}$.

Appendix A

Appendix Title Here

Write your Appendix content here.