University of California

Los Angeles

First-Order Methods for Trace Norm Minimization

A thesis submitted in partial satisfaction

of the requirements for the degree

Master of Science in Electrical Engineering

by

Hsiao-Han Chao

2013

Abstract of the Thesis

First-Order Methods for Trace Norm Minimization

by

Hsiao-Han Chao

Master of Science in Electrical Engineering

University of California, Los Angeles, 2013

Professor Lieven Vandenberghe, Chair

Minimizing the trace norm (sum of singular values) of a matrix has become popular as a convex heuristic for computing low rank approximations, with numerous applications in control, system theory, statistics, and machine learning. However, it is usually too expensive to solve these matrix optimization problems with second-order methods, such as the interior-point method, given that the scale of the problems is relatively large. In this thesis, we compare several first-order methods for nondifferentiable convex optimization based on splitting algorithms, and apply them to primal, dual, or primal-dual optimality conditions. The implementation aspects of the algorithms are discussed in detail and their performance is compared in experiments with randomly generated data as well as system identification test sets. We show that on large-scale problems, several of the first-order methods reach a modest accuracy within a shorter time than the interior-point solvers. Based on the experiments, the most promising methods are the Alternating Direction Method of Multipliers and the Pock-Chambolle semi-implicit algorithm.

The thesis of Hsiao-Han Chao is approved.

University of California, Los Angeles

2013

To my dear family and family in Christ

Table of Contents

List of Figures

List of Tables

Acknowledgments

This thesis could never have come together without the support and guidance of Professor Lieven Vandenberghe, to whom I owe countless thanks and deep gratitude. I am thankful for having been showered by his elegant and vigorous teaching, and his great characters both as a teacher and as an advisor. His passion and insight for research has been inspiring, yet he is ever encouraging and patient with my slowness. Indeed, I would say, God has provided the best advisor I could ever have to enjoy working with and learning from, that I never expected before coming to UCLA.

I would also like to thank Professor Alan Laub and Professor Vwani Roychowdhury, who have kindly agreed to be on my committee, and have enriched me intellectually through their classes.

I am grateful for the support and friendship of our group members: Daniel, Jinchao, Rong, and Yifan, who have been so welcoming and encouraging, always passionately talking about math and research, and of course, fun stuffs. Many thanks go to my friends, roommates, and neighbors, who have definitely colored my life here and made me feel like home. Thanks to Grace on Campus and Grace Community Church, where the Bible is being preached clearly and faithfully.

Thanks to Dr. Lu, who graciously supported me for one quarter, and has been willing to get to know and share encouragements with me. Special thanks to my dear family, who has fully supported me through love, care, provision and prayer.

Last but not least, let all thanks be to God, who has sustained me through all the up-and-downs and has shown to be faithful. I know there is way more to learn and way longer to go, but praises be to Him who has allowed us to rejoice always in His light, in His kingdom to come, into which we can enter only through the Lord Jesus Christ our Savior, but none of our own merits.

CHAPTER 1

Introduction

Although there are relatively well-developed methods for solving convex optimization problems, a lot of problems in practice are not convex by nature. One of the common approaches is to approximate the nonconvex problem with a convex model, or to use convex optimization as heuristics of finding promising solutions. In this thesis we minimize the trace norm (sum of singular values) of a matrix as a heuristic for finding low-rank solutions. This was proposed by Fazel et al. in 2001 [FHB01], motivated by the observation that the resulting matrix from trace norm minimization often has low rank, and it can be viewed as a generalization of ℓ_1-norm minimization techniques for cardinality minimization.

1.1 Rank Minimization

The rank of an appropriately constructed matrix could be representing the order, dimension, or complexity of an underlying system from various contexts. System identification or model estimation problem can be expressed as minimizing the rank of a matrix subject to constraints, because we believe there is a rather simple governing model underlying observations, as noisy or interfered by unknown factors. A general formulation is

$$\begin{array}{ll} \text{minimize} & \mathbf{rank}(A(x) - B) \\ \text{subject to} & \mathcal{F}(x) \in C, \end{array}$$

where $A(x) - B$ is an affine matrix-valued function of x. This rank minimization problem arises in control, signal processing, and statistics. Particularly, there has been growing interest in robotics, system identification, and machine learning. However, while special purpose solvers exist, the problem is NP-hard in general.

In this thesis we consider a general approximate problem of minimizing the quadratically regularized trace norm of the affine matrix-valued function, which has been shown successful in several applications, such as the minimum-rank matrix completion problem. There has been a proven special case where trace norm minimization gives solutions to rank minimization [RFP10].

Most of these problems that arise in practice are relatively large-scale, which makes it impractical to solve with second-order methods (Newton's method or interior-point method), considering the memory capacity and computational power of today's machines. Thus, more attention has been drawn to a class of first-order methods [MGC11, FPS11], especially those based on splitting techniques.

1.2 System Identification

This thesis is motivated by applications in system identification. We follow the idea presented in [LV09] that shows how trace norm minimization is utilized as a variant in the earlier subspace algorithm. A discrete-time linear time-invariant state-space model has the form

$$\begin{aligned} x(t) &= Ax(t-1) + Bu(t-1) \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{1.1}$$

where $u(t) \in \mathbf{R}^l, y(t) \in \mathbf{R}^k$ are input and output sequences that can be measured for time $t = 0, \ldots, N$, and $x(t) \in \mathbf{R}^m$ is the state variable at t. The goal is to estimate the model order m, the matrices A, B, C, D, and the initial state $x(0)$.

If we define block Hankel matrices

$$Y = \begin{pmatrix} y(0) & y(1) & y(2) & \cdots & y(N-r) \\ y(1) & y(2) & y(3) & \cdots & y(N-r+1) \\ \vdots & \vdots & \vdots & & \vdots \\ y(r) & y(r+1) & y(r+2) & \cdots & y(N) \end{pmatrix} \in \mathbf{R}^{k(r+1)\times(N-r+1)}$$

and

$$U = \begin{pmatrix} u(0) & u(1) & u(2) & \cdots & u(N-r) \\ u(1) & u(2) & u(3) & \cdots & u(N-r+1) \\ \vdots & \vdots & \vdots & & \vdots \\ u(r) & u(r+1) & u(r+2) & \cdots & u(N) \end{pmatrix} \in \mathbf{R}^{l(r+1)\times(N-r+1)},$$

then (1.1) can be written as

$$Y = JX + KU, \tag{1.2}$$

where

$$J = \begin{pmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^r \end{pmatrix}, \quad K = \begin{pmatrix} D & 0 & 0 & \cdots & 0 \\ CB & D & 0 & \cdots & 0 \\ CAB & CB & D & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ CA^{r-1}B & CA^{r-2}B & CA^{r-3}B & \cdots & D \end{pmatrix},$$

and

$$X = \begin{pmatrix} x(0) & x(1) & x(2) & \cdots & x(N-r) \end{pmatrix}.$$

We are given the matrices U and Y, or noisy measurements of them, where r is a chosen parameter such that $r > m$ is ensured,and J, K, X are unknown. We refer the reader to [Lju99, chapter 10], [VV07, chapter 9] for more details.

Subspace Algorithm

Taking a matrix $U^{\perp}$ whose columns span the nullspace of U and multiplying (1.2) on the right with it, we have

$$YU^{\perp} = JXU^{\perp}. \tag{1.3}$$

Assuming no cancellation of rank in the matrix multiplication $XU^\perp$ and $\mathbf{rank}(X) = m$, which hold generally with randomly chosen input sequence, then $\mathbf{rank}(YU^\perp) = m$ with exact measurements of input and output.

A system realization can therefore be determined as follows. First compute a matrix $\tilde{J}$ whose columns form a basis for the column space of $YU^\perp$, and then we have $\tilde{J} = JT$, where T is a nonsingular matrix. Therefore,

$$\tilde{J} = \begin{pmatrix} \tilde{J}_0 \\ \tilde{J}_1 \\ \vdots \\ \tilde{J}_r \end{pmatrix} = \begin{pmatrix} C \\ CA \\ \vdots \\ CA^r \end{pmatrix} T$$

is the extended observability matrix for the system

$$\begin{aligned} \tilde{x}(t) &= \tilde{A}\tilde{x}(t-1) + \tilde{B}u(t-1) \\ y(t) &= \tilde{C}\tilde{x}(t) + \tilde{D}u(t), \end{aligned}$$

where $\tilde{A} = T^{-1}AT,\ \tilde{B} = T^{-1}B,\ \tilde{C} = CT$, and $\tilde{D} = D$, which can be verified as equivalent to (1.1), with the only difference being a change of coordinates of the state space, $x(t) = T\tilde{x}(t)$. We then have $\tilde{C} = \tilde{J}_0$, and $\tilde{A}$ through solving

$$\begin{pmatrix} \tilde{J}_1 \\ \tilde{J}_2 \\ \vdots \\ \tilde{J}_r \end{pmatrix} = \begin{pmatrix} \tilde{J}_0 \\ \tilde{J}_1 \\ \vdots \\ \tilde{J}_{r-1} \end{pmatrix} \tilde{A}. \tag{1.4}$$

Following from the linear equations

$$\tilde{C}\tilde{A}^t\tilde{x}(0) + \sum_{s=0}^{t-1} \tilde{C}\tilde{A}^{t-s}\tilde{B}u(s) + \tilde{D}u(t) = y(t), \quad t = 0, \ldots, N, \tag{1.5}$$

the matrices $\tilde{B}, \tilde{D}$ and the initial state $\tilde{x}(0)$ can be computed.

If the input or output data are with error, then the above equations would not be solvable. Instead, one can estimate the system order m using thresholding on the singular values, and get an estimate of $\tilde{J}$ by keeping only the first m terms in the singular value decomposition. Solving for least-square solutions for the overdetermined equations (1.4) and (1.5) then gives us estimates of the state-space model.

Identification via Trace Norm Minimization

Alternatively, if we denote the measured outputs as $y_{\text{meas}}(t)$, and obtain $y(t), t = 0, \ldots, N$ by solving the problem

$$\text{minimize} \quad \|YU^{\perp}\|_* + \gamma \sum_{t=0}^{N} \|y(t) - y_{\text{meas}}(t)\|_2^2 \tag{1.6}$$

for a weighting parameter $\gamma > 0$ on the quadratic regularization term. With the optimal solution $y(t)$ of (1.6) with a selected γ that gives desirable rank property and error on the trade-off curve, we can compute $YU^{\perp}$ and solve for the estimates of the state-space model as described in the previous section. This alternative provides the advantage of preserving linear matrix structure in the low-rank approximation. One simple variation is to use different weights for each measurement,

$$\text{minimize} \quad \|YU^{\perp}\|_* + \sum_{t=0}^{N} \|W_t(y(t) - y_{\text{meas}}(t))\|_2^2.$$

This approach to system identification is promising because it is easy to impose additional constraints according to prior knowledge, to add regularization terms on model inputs and outputs, or to accommodate problems with missing data points [LHV13, DSC07, GJM09].

1.3 Outline

Chapter 2 provides necessary background for convex optimization and preliminaries for understanding derivation of numerical methods, and introduces the quadratically regularized trace norm minimization problem that the thesis is focusing on. The first-order numerical methods and their convergence are discussed in chapter 3, and how they are applied on trace norm minimization is described in chapter 4. Finally, we show the experiments in chapter 5.

CHAPTER 2

Convex Optimization

In this chapter we introduce the basic knowledge of convex optimization necessary for understanding the work of the thesis. Consider a convex optimization problem of the following form,

$$\text{minimize} \quad f(x) + g(A(x)), \tag{2.1}$$

where f and g are convex functions and A is a linear operator. All convex optimization problems can be represented in this form. For example,a constrained optimization problem,

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & A(x) \in C, \end{array}$$

where C is a convex set, can be represented as

$$\text{minimize} \quad f(x) + I_C(A(x)), \tag{2.2}$$

where I_C is the indicator function of C such that $I_C(y)$ equals zero if $y \in C$, infinity if $y \notin C$. Another commonly encountered example is to have a norm function as the objective function,

$$\text{minimize} \quad f(x) + \|A(x) - B\|, \tag{2.3}$$

where $\|\cdot\|$ can be any norm, for example, the ℓ_∞-norm or ℓ_1-norm for a vector, and the spectral norm or trace norm for a matrix.

We denote the value of the objective function of (2.1) as p, and its minimum p^*, which is $-\infty$ if the problem is unbounded below, ∞ if it is infeasible. When the optimal value is attained, x^* denotes a solution that gives the value p^*.

2.1 Dual Problem

We consider the Lagrangian dual of our problem, and refer to problem (2.1) as the primal problem. In order to have a useful Lagrangian, we first reformulate the primal problem as

$$\begin{array}{ll} \text{minimize} & f(x) + g(y) \\ \text{subject to} & A(x) = y. \end{array} \tag{2.4}$$

The Lagrangian is then defined with multiplier z as

$$L(x, y, z) = f(x) + g(y) + \langle z, A(x) - y \rangle, \tag{2.5}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. We obtain the dual objective function by taking the infimum of the Lagrangian over the primal variables.

$$\begin{aligned} \inf_{x,y} L(x, y, z) &= \inf_x (f(x) + \langle A_{\text{adj}}(z), x \rangle) + \inf_y (g(y) - \langle z, y \rangle) \\ &= -\sup_x (-\langle A_{\text{adj}}(z), x \rangle - f(x)) - \sup_y (\langle z, y \rangle - g(y)) \\ &= -f^*(-A_{\text{adj}}(z)) - g^*(z), \end{aligned}$$

where A_{adj} is the adjoint of A, such that $\langle z, A(x) \rangle = \langle A_{\text{adj}}(z), x \rangle$ for all x and z, and in the last line we use the conjugate function notation.

Definition 2.1 (Conjugate function) *The conjugate function f^* of a function f is defined as*

$$f^*(y) = \sup_{x \in \mathbf{dom} f} (\langle y, x \rangle - f(x)).$$

The conjugate function f^* is always convex regardless of the convexity of f.

The dual problem is then defined as the maximization of the dual function,

$$\text{maximize} \quad -f^*(-A_{\text{adj}}(z)) - g^*(z). \tag{2.6}$$

Note that this is maximization over a concave function, which is also a convex optimization problem. We denote the value of the dual objective function as d, and its maximum d^*, which is ∞ if the problem is unbounded above, $-\infty$ if infeasible. If the interior of the feasible set of either the primal or dual problem is nonempty, strong duality ensures that $p^* = d^*$, otherwise we always have the weak duality, $p^* \geq d^*$.

The conjugate function of I_C is the support function of C, defined as $h_C(z) = \sup_{x \in C}\langle z, x\rangle$, and thus the dual of (2.2) is

$$\text{maximize} \quad -f^*(-A_{\text{adj}}(z)) - h_C(z).$$

The conjugate function of a norm is the indicator function of its dual unit norm ball, therefore the dual of (2.3) is

$$\text{maximize} \quad -f^*(-A_{\text{adj}}(z)) - I_{\|\cdot\|_{\text{d}} \leq 1}(z).$$

2.2 Optimality Condition

Since the functions we are dealing with are not always differentiable, here we introduce the subdifferential of a function.

Definition 2.2 (Subdifferential) *A **subgradient** h of a function f at a point $x \in \mathbf{dom} f$ is a vector that satisfies*

$$f(y) \geq f(x) + \langle h, y - x\rangle \ \forall y \in \mathbf{dom} f.$$

The subdifferential of f at x is the set of all subgradients at x, denoted as

$$\partial f(x) = \{h | f(y) \geq f(x) + \langle h, y - x\rangle \ \forall y \in \mathbf{dom} f\}.$$

For a convex function f, the subdifferential ∂f is always nonempty, except possibly at the boundary of $\mathbf{dom} f$. For example, $f(x) = -\sqrt{x}$ is not subdifferentiable at $x = 0$. If a convex function f is differentiable at point x, meaning the gradient, $\nabla f(x)$ exists, then $\partial f(x) = \{\nabla f(x)\}$. If zero is in the subdifferential set at x, then $f(x)$ is a global minimum of the function f, since $f(y) \geq f(x) \ \forall y \in \mathbf{dom} f$.

The primal objective function in (2.1) is at its minimum when zero is in its subdifferential,

$$0 \in \partial f(x^*) + A_{\text{adj}}(\partial g(A(x^*))), \tag{2.7}$$

where $A_{\text{adj}}(\partial g(y)) = \{A_{\text{adj}}(h) | h \in \partial g(y)\}$. This inclusion is the primal optimality condition. The dual optimality condition can be obtained similarly,

$$0 \in -A(\partial f^*(-A_{\text{adj}}(z^*))) + \partial g^*(z^*), \tag{2.8}$$

by noting that the problem (2.6) is equivalent to solving the minimization of the negative of its objective function, which is then convex. If we reformulate (2.7) as

$$\begin{aligned} 0 &\in \partial f(x^*) + A_{\text{adj}}(z^*) \\ z^* &\in \partial g(A(x^*)), \end{aligned}$$

and note that $(\partial g)^{-1} = \partial g^*$ for g being a convex and closed function, it is equivalent to

$$\begin{aligned} 0 &\in A_{\text{adj}}(z^*) + \partial f(x^*) \\ 0 &\in -A(x^*) + \partial g^*(z^*), \end{aligned} \tag{2.9}$$

which is usually identified as the primal-dual optimality condition. For more detailed theories related to convex optimization, please refer to [BV04].

2.3 Monotone Operators and Resolvents

We now introduce a category of operators which maps a vector $x \in \mathbf{R}^m$ to a set $F(x) \in \mathbf{R}^m$. Specifically, in this thesis the operators we deal with are monotone operators.

Definition 2.3 (Monotone operator) *An operator F is monotone if it satisfies*

$$\langle y - \hat{y}, x - \hat{x} \rangle \geq 0, \ \forall x, \hat{x} \in \mathbf{dom} F, y \in F(x), \hat{y} \in F(\hat{x}).$$

Furthermore, F is maximal monotone if and only if $\mathbf{Im}(I + F) = \mathbf{R}^m$.

This means that if F is maximal monotone, the image of the operator $(I + F)$ is the whole space, and hence there exists a solution x to $y \in x + F(x)$ for all $y \in \mathbf{R}^m$.

The subdifferential ∂f of a convex function f is a monotone operator. If f is closed and convex, then ∂f is maximal monotone, and the inverse of it is the subdifferential of the conjugate of f, $(\partial f)^{-1} = \partial f^*$, which we used in the previous section to derive the primal-dual optimality condition. Another example of a maximal monotone operator is a skew symmetric linear operator

$$F = \begin{pmatrix} 0 & \hat{A}^T \\ -\hat{A} & 0 \end{pmatrix},$$

where $\hat{A}$ is any matrix. A nonnegative combination of two maximal monotone operators is also maximal monotone, provided that the intersection of their domain has nonempty interior.

The resolvent operator of an operator is also often used in the thesis.

Definition 2.4 (Resolvent) *The resolvent of an operator F is defined as*

$$(I + tF)^{-1},$$

where I is the identity matrix and $t > 0$ is a scalar.

An operator is monotone if and only if its resolvent is firmly nonexpansive, such that for all $y, \hat{y} \in \mathbf{dom}(I + tF)^{-1}, x \in (I + tF)^{-1}y, \hat{x} \in (I + tF)^{-1}\hat{y}$,

$$\langle x - \hat{x}, y - \hat{y} \rangle \geq \|x - \hat{x}\|_2^2.$$

This implies the uniqueness of resolvent, and thus we can write $x = (I + tF)^{-1}(y)$ instead of the inclusion. If further F is maximal monotone, then $(I + tF)^{-1}(y)$ exists for every y, since $\mathbf{dom}(I + tF)^{-1} = \mathbf{R}^m$. We use the following two expressions interchangeably,

$$x = (I + tF)^{-1}y \iff \frac{y - x}{t} \in F(x).$$

The resolvent of ∂f is the proximal mapping.

Definition 2.5 (Proximal operator) *The proximal operator of a closed and convex function f is defined as*

$$\begin{aligned}\mathbf{prox}_{tf}(y) &= (I + t\partial f)^{-1}(y) \\ &= \mathrm{argmin}_x(f(x) + \tfrac{1}{2t}\|x - y\|_2^2),\end{aligned}$$

since the optimality condition of the minimization is $0 \in \partial f(x) + \frac{1}{t}(x - y)$.

The proximal operator of the indicator function of a closed and convex set C is the Euclidean projection on C.

$$\begin{aligned}\mathbf{prox}_{tI_C}(y) &= (I + tN_C)^{-1}(y) \\ &= \mathrm{argmin}_x(I_C(x) + \tfrac{1}{2t}\|x - y\|_2^2) \quad = \quad P_C(y),\end{aligned}$$

where $N_C = \partial I_C$ is the normal cone to C.

The resolvent of a skew symmetric linear operator is

$$\begin{pmatrix} I & t\hat{A}^T \\ -t\hat{A} & I \end{pmatrix}^{-1},$$

whose existence and uniqueness can be easily identified.

For more complete discussion about these operators, please refer to [Roc76, Roc97].

2.4 Trace Norm Minimization Problems

In this section we introduce the specific convex optimization problem we are solving in the form of the previous sections. Then we simplify the expressions by using a vectorized notation. Consider a quadratically regularized trace norm minimization problem of the following form,

$$\text{minimize} \quad \tfrac{1}{2}x^T Px + \|A(x) - B\|_* \ , \qquad (2.10)$$

where $A(x) = A_1x_1 + A_2x_2 + \cdots + A_nx_n$. Here $x \in \mathbf{R}^n$ is the optimization variable, and $P \in \mathbf{S}^n_+$, A_i, $B \in \mathbf{R}^{p\times q}$,where $i = 1, \ldots, n$, are problem parameters. The trace norm of a matrix $Y \in \mathbf{R}^{p\times q}$ is the sum of singular values of Y, denoted $\|Y\|_*$, and is also known as the nuclear norm, the Schatten 1-norm, and the Ky Fan $\min(p, q)$-norm [HJ85, chapter 3] [HJ91, chapter 7]. We can put (2.10) into the form of (2.1) by taking

$$\begin{array}{rcl} f(x) & = & \frac{1}{2}x^TPx \\ g(Y) & = & \|Y - B\|_*, \end{array}$$

which are both convex.

2.4.1 Dual Problem and Optimality Condition

To obtain the dual form (2.6) of our problem, we need f^*, g^*, and A_{adj}. We have

$$f^*(y) = \sup_x (y^Tx - \frac{1}{2}x^TPx) = \frac{1}{2}y^TP^\dagger y,$$

with $\mathbf{dom} f^* = \mathbf{Range}(P)$. In this thesis we use the Moore-Penrose pseudoinverse of P, denoted as $P^\dagger$, such that $P^\dagger = P^{-1}$ if P is nonsingular, and it has the property $P^\dagger PP^\dagger = P^\dagger$. Particularly, if $P = \sum_i \lambda_i q_i q_i^T$ is an eigenvalue decomposition of P, then

$$P^\dagger = \sum_{\lambda_i>0} \frac{1}{\lambda_i} q_i q_i^T.$$

Note that P is usually diagonal in our applications.

For the conjugate of $g(Y) = \|Y - B\|_*$, follow the definition and that the spectral norm (largest singular value) is the dual of the trace norm,

$$\begin{array}{rcl} g^*(Z) & = & \sup_Y (\mathbf{Tr}(Z^TY) - \|Y - B\|_*) \\ & = & \mathbf{Tr}(B^TZ) + I_{\|\cdot\|_2\le 1}(Z). \end{array}$$

To derive A_{adj}, using the relation $\langle Z, A(x)\rangle = \langle A_{\text{adj}}(Z), x\rangle$, and comparing that

$$\begin{aligned}
\langle Z, A(x)\rangle &= \mathbf{Tr}(Z^T A(x)) \\
&= \sum_{i=1}^{n} x_i \mathbf{Tr}(Z^T A_i) \\
\langle A_{\text{adj}}(Z), x\rangle &= A_{\text{adj}}(Z)^T x,
\end{aligned}$$

it is clear that $A_{\text{adj}}(Z) = (\mathbf{Tr}(Z^T A_1), \mathbf{Tr}(Z^T A_2), \ldots, \mathbf{Tr}(Z^T A_n))$.

The dual problem of (2.10) is then

$$\begin{array}{ll}
\text{maximize} & -\frac{1}{2}A_{\text{adj}}(Z)^T P^{\dagger} A_{\text{adj}}(Z) - \mathbf{Tr}(B^T Z) - I_{\|\cdot\|_2 \le 1}(Z) \\
\text{subject to} & A_{\text{adj}}(Z) \in \mathbf{Range}(P).
\end{array} \tag{2.11}$$

Moving on to the optimality condition, we have

$$0 \in Px^* + A_{\text{adj}}(\partial \|A(x^*) - B\|_*) \tag{2.12}$$

for the primal, where

$$\partial \|Y\|_* = \{UV^T + W |\ Y = U\mathbf{diag}(s)V^T, U^T W = 0, WV = 0, \|W\|_2 \le 1\}.$$

Here the compact form of singular value decomposition of Y is used, and W has orthogonal column and row spaces with Y [RFP10]. The dual optimality condition is

$$0 \in A(P^{\dagger} A_{\text{adj}}(Z^*)) + B + N_{\|\cdot\|_2 \le 1}(Z^*). \tag{2.13}$$

As you may see, the linear operators A and A_{adj} that perform transformation between vectors and matrices seem rather complicated. Hence in the following section, we redefine the problem using vectorized notations for the convenience of formulating the algorithms in later chapters, and we will be using these notations henceforth in the thesis.

2.4.2 Vectorized Notations

We use $z = \mathbf{vec}(Z)$ to denote a vector $z \in \mathbf{R}^{pq}$ formed by appending all the columns of $Z \in \mathbf{R}^{p \times q}$ one after another into a single column vector, and $Z =$

$\mathbf{mat}(z)$ represents the inverse of this process. We also use the linear operator $A(x) = \hat{A}x$, where

$$\hat{A} = (\mathbf{vec}(A_1)\ \mathbf{vec}(A_2)\ \cdots\ \mathbf{vec}(A_n)) \in \mathbf{R}^{pq \times n},$$

and thus $A_{\mathrm{adj}}(z) = \hat{A}^T z$.

Using the newly defined notations, our problem is expressed as

$$\text{minimize} \quad f(x) + g(\hat{A}x), \tag{2.14}$$

where f is as defined earlier in the section, and we overload the function g so that

$$g(y) = \|\mathbf{mat}(y) - B\|_*.$$

We then have

$$g^*(z) = \mathbf{vec}(B)^T z + I_C(z),$$

where the set C denotes a set of vectors in $\mathbf{R}^{pq}$ such that the spectral norm of the corresponding matrix form in $\mathbf{R}^{p \times q}$ is less than or equal to one,

$$C = \{z \mid \|\mathbf{mat}(z)\|_2 \leq 1\}. \tag{2.15}$$

The dual problem can be written as

$$\text{maximize} \quad -f^*(\hat{A}^T z) - g^*(z). \tag{2.16}$$

Noting also that

$$\begin{array}{rcl} \partial f(x) & = & Px \\ \partial f^*(w) & = & P^\dagger w \\ \partial g(y) & = & \mathbf{vec}(\partial\|\mathbf{mat}(y) - B\|_*) \\ \partial g^*(z) & = & \mathbf{vec}(B) + N_C(z), \end{array}$$

we then write the primal optimality condition as

$$0 \in \partial f(x) + \hat{A}^T \partial g(\hat{A}x), \tag{2.17}$$

and the dual optimality condition,

$$0 \in -\hat{A}\partial f^*(-\hat{A}^T z) + \partial g^*(z). \tag{2.18}$$

The primal-dual optimality condition can now be written as one inclusion,

$$0 \in \begin{pmatrix} 0 & \hat{A}^T \\ -\hat{A} & 0 \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} + \begin{pmatrix} \partial f(x) \\ \partial g^*(z) \end{pmatrix}. \tag{2.19}$$

CHAPTER 3

First-Order Splitting Methods for Convex Optimization

In this chapter we introduce several numerical methods for solving a general convex optimization problem, with possibly some assumptions on the objective function and its domain. Specifically, we focus on first-order splitting methods since our application is on large-scale data. Second-order methods usually achieve higher accuracy, but become inefficient as the size of the problem grows larger.

Consider a maximal monotone zero inclusion problem,

$$0 \in T(x),$$

whose solutions can be easily shown to be the fixed points of the resolvent $(I + tT)^{-1}$. The fixed-point iteration

$$x^+ = (I + tT)^{-1}x \tag{3.1}$$

is called the proximal point algorithm, which converges independently of the choice of t [Roc76]. Here x^+ represents the updated value of the current iterate x. (The same notation will be used in the following sections.) Equation (3.1) is equivalent to

$$\frac{x^+ - x}{t} \in -T(x^+),$$

which can be interpreted as the backward Euler discretization of the differential inclusion,

$$\frac{dx(t)}{dt} \in -T(x(t)).$$

On the other hand, the forward Euler discretization computes x^+ from $x^+ \in x - tT(x)$, which is cheaper but requires $T(x)$ to be single-valued and converges only if the step size t is sufficiently small.

If we apply the proximal point algorithm, for example, to the dual optimality condition (2.8), we have

$$\frac{z - z^+}{t} \in T(z^+) = -\hat{A}\partial f^*(-\hat{A}^T z^+) + \partial g^*(z^+).$$

Using the inverse property such that $(\partial f^*)^{-1} = \partial f$ and $(\partial g^*)^{-1} = \partial g$ gives an equivalent formulation,

$$0 \in \begin{pmatrix} 0 & 0 & \hat{A}^T \\ 0 & 0 & -I \\ -\hat{A} & I & 0 \end{pmatrix} \begin{pmatrix} x^+ \\ y^+ \\ z^+ \end{pmatrix} + \begin{pmatrix} \partial f(x^+) \\ \partial g(y^+) \\ 0 \end{pmatrix} + \frac{1}{t} \begin{pmatrix} 0 \\ 0 \\ z^+ - z \end{pmatrix},$$

which is the optimality condition of minimizing

$$\tilde{L}_t(x^+, y^+, z) = f(x^+) + g(y^+) + \langle z, \hat{A}x^+ - y^+ \rangle + \frac{t}{2}\|\hat{A}x^+ - y^+\|_2^2,$$

jointly over x^+ and y^+, and taking $z^+ = z + t(\hat{A}x^+ - y^+)$. We refer to $\tilde{L}_t(x, y, z)$ as an augmented Lagrangian of (2.4). However, the minimization may be too expensive or too difficult to accomplish.

The first-order methods introduced in this chapter all seek to incorporate fully or partially the backward Euler rule due to its desirable numerical stability and accuracy, while various tricks are used in order to reduce the cost of calculation.

3.1 Douglas-Rachford Splitting

Douglas-Rachford splitting is a method for finding a zero for the sum of two maximal monotone operators,

$$0 \in F(x) + G(x). \tag{3.2}$$

The method does not require the evaluation of the resolvent of $F + G$; instead, it requires resolvents of the two operators individually, which hopefully are less expensive to evaluate. While the idea was first introduced for the discretized heat equation in mid-1950's by Douglas and Rachford [DR56], it got its name as the Douglas-Rachford scheme in [LM79]. It is later shown to be a special case of the proximal point algorithm applied to a specific splitting operator [EB92].

Algorithm 1 (Douglas-Rachford Splitting)

```
***Initialization*** *choose any $\tilde{x}$.*

***Iteration*** *repeat*
    1. $x^+ = (I + tG)^{-1}(\tilde{x})$
    2. $\bar{x} = (I + tF)^{-1}(2x^+ - \tilde{x})$
    3. $\tilde{x}^+ = \tilde{x} + \bar{x} - x^+$
```

According to [EB92], x converges to some zero of $F + G$ if the solution set is not empty, and there is no assumption on the step size t.

3.1.1 ADMM

We may apply Douglas-Rachford splitting on any of the optimality conditions, depending on which one of an optimization problem is less expensive to solve. Particularly, if we apply (**Alg. 1**) to the dual optimality condition (2.8) and work out the mathematics, it is shown by [Gab83] to be equivalent to the Alternating Direction Method of Multipliers, often addressed as ADMM.

ADMM in itself is a popular first-order method especially for large-scale applications where high accuracy is not essential. It usually has a fast convergence to a modest accuracy, but a slow convergence to a higher one. The history of its development can be found in [EB92], where its origin is dated back to the mid-1970's.

Recall the example of the proximal point method in the beginning of the chapter; ADMM also seeks to minimize the augmented Lagrangian $\tilde{L}_t(x, y, z)$, but alternately with respect to x and y, which is how it got its name.

Algorithm 2 (ADMM)

```
    ***Initialization*** *choose any $y$ and $z$.*

    ***Iteration*** *repeat*
        1. $x^+ = \text{argmin}_{\bar{x}} \tilde{L}_t(\bar{x}, y, z)$
        2. $y^+ = \text{argmin}_{\bar{y}} \tilde{L}_t(x^+, \bar{y}, z)$
               $= \mathbf{prox}_{g/t}(\hat{A}x^+ + z/t)$
        3. $z^+ = z + t(\hat{A}x^+ - y^+)$
```

The only difference is that the steps in (**Alg. 2**) does not minimize both variables simultaneously.

As with Douglas-Rachford splitting method, convergence of ADMM is independent of the choice of t, which can be either a fixed value or determined adaptively in each iteration.

3.2 Forward-Backward Splitting

Consider again the zero inclusion problem (3.2) and the differential inclusion

$$\frac{dx(t)}{dt} \in -F(x(t)) - G(x(t)).$$

Besides the forward Euler rule and the backward Euler rule, the discretization can also be done by applying the mixture of the two, which we call the forward-backward discretization,

$$\frac{x^+ - x}{t} \in -F(x) - G(x^+).$$

Solving for x^+ in the this case gives the forward-backward update,

$$x^+ = (I + tG)^{-1}(x - tF(x)).$$

We can see that this requires F to be single-valued and that $\mathbf{dom}G \subseteq \mathbf{dom}F$.

Given a simple optimization problem whose objective function can be represented in two parts,

$$\text{minimize} \quad f(x) + g(x), \tag{3.3}$$

by applying forward-backward Euler update on its optimality condition, we obtain the proximal gradient method.

Compared to Douglas-Rachford splitting method, forward-backward method requires only one resolvent instead of two, which provides a potential advantage for decomposing the "dense" part into F. However, it also requires a stronger condition for convergence. A convergence proof of the forward-backward method for co-coercive operator F is given in [Tse91]. Co-coercivity is such that for all $x, \hat{x} \in \mathbf{dom}F$, $y \in F(x), \hat{y} \in F(\hat{x})$, there exists some $\gamma > 0$ that,

$$(y - \hat{y})^T(x - \hat{x}) \geq \gamma \|y - \hat{y}\|_2^2.$$

Assumption of a nonempty solution set is required, and the step size must be chosen as $t \in (0, 2\gamma)$.

3.2.1 FISTA

FISTA is the acronym for Fast Iterative Shrinkage-Thresholding Algorithm as introduced by Beck and Teboulle [BT09]. It is a proximal gradient method accelerated by performing the proximal gradient at an extrapolation point of the previous two points.

Consider again problem (3.3), if f is convex and differentiable with $\mathbf{dom}f = \mathbf{R}^n$, and g is closed and convex so that its proximal operator is well defined, and preferably inexpensive to compute, then FISTA can be applied to solve the problem. A simple version of FISTA is described in the following algorithm.

Algorithm 3 (FISTA)

Initialization *choose any* $x = x^-$.

Iteration *for* $k \geq 1$, *repeat*

$$\begin{array}{lll} 1. \quad \bar{x} & = & x + \frac{k-2}{k+1}(x - x^-) \\ 2. \quad x^+ & = & \mathbf{prox}_{tg}(\bar{x} - t\nabla f(\bar{x})) \end{array}$$

Here k is the iteration count, and x^- from two iterations ago. $\{x^-, x, x^+\}$ are consecutive elements from the sequence $\{x_k\}$ as generated by the iterations of the algorithm. Same idea applies in chapter 4.

In each iteration, $\bar{x}$ might not be a feasible point, but x^+ always is. FISTA is not a descent method, although we can surely make it one through some variation of the update of x^+. Convergence is guaranteed if the optimal value p^* is finite and attained at some point x^*, and ∇f is Lipschitz continuous with constant L,

$$\|\nabla f(x) - \nabla f(\hat{x})\|_2 \leq L\|x - \hat{x}\|_2 \ \forall x, \hat{x}.$$

The step size t can be either fixed to $1/L$ or determined by a line search. The convergence analysis requires that

$$f(x^+) \leq f(\bar{x}) + \nabla f(\bar{x})^T(x^+ - \bar{x}) + \frac{1}{2t}\|x^+ - \bar{x}\|_2^2,$$

which is known to hold for $t \in (0, \frac{1}{L}]$. Hence, we can start with a larger step size $t = t_0 > 1/L$, with a parameter $\beta \in (0, 1)$. In each iteration, replace step 2 with the following steps.

```
$x^+ = \mathbf{prox}_{tg}(\bar{x} - t\nabla f(\bar{x}))$
while $f(x^+) > f(\bar{x}) + \nabla f(\bar{x})^T(x^+ - \bar{x}) + \frac{1}{2t}\|x^+ - \bar{x}\|_2^2$
    $t^+ = \beta t$
    $x^+ = \mathbf{prox}_{t^+g}(\bar{x} - t^+\nabla f(\bar{x}))$
end
```

The t^+ in the inner while-loop is the update for each iteration of the inner loop. Once t becomes less than or equal to $1/L$, Lipschitz continuity of ∇f guarantees that the while-loop condition will no longer be met.

The rate of convergence in function values is proven to be $\mathcal{O}(1/k^2)$ according to [BT09], an improvement from the worst-case convergence rate $\mathcal{O}(1/k)$ of proximal gradient methods, which does not have the extrapolation step but costs about the same with FISTA in each iteration. According to [BT09] and their experiment on image deblurring, FISTA works well both theoretically and in practice.

3.2.2 Tseng's Modified Forward-Backward Splitting

Tseng presented a modified forward-backward method in [Tse00] that uses an approximate backward Euler rule,

$$\frac{x^+ - x}{t} \in -F(\bar{x}) - G(\bar{x}),$$

where $\bar{x}$ is obtained through a forward-backward update,

$$\frac{\bar{x} - x}{t} \in -F(x) - G(\bar{x}).$$

The method is described in the following algorithm.

Algorithm 4 (Tseng's Modified Forward-Backward Splitting)

Initialization *choose any* $x \in \mathbf{dom}F$.

Iteration *repeat*

$$\begin{aligned} 1.\quad \bar{x} &= (I+tG)^{-1}(x - tF(x)) \\ 2.\quad x^+ &= \bar{x} - t(F(\bar{x}) - F(x)) \end{aligned}$$

Here we need to assume $\mathbf{dom}F = \mathbf{R}^n$ to ensure that $x^+ \in \mathbf{dom}F$. Convergence requires Lipschitz continuity of F, which is implied by, but usually less constrained than co-coercivity. The step size can be fixed as $t = 1/L$ or determined by a backtracking line search. The line search is done similarly to that of FISTA. We start with $t = t_0 > 1/L$, and multiply it by β whenever the following condition is not satisfied.

$$t\|F(\bar{x}) - F(x)\|_2 \leq \theta\|\bar{x} - x\|_2$$

When $t \leq \theta/L$, the condition is guaranteed to be satisfied, and $\theta \in (0,1)$ is an algorithm parameter.

3.3 Semi-Implicit Methods

Semi-implicit method describes yet another way to compute the updates. A natural consideration would be the primal-dual optimality condition (2.19).

$$\frac{1}{t}\begin{pmatrix} x - x^+ \\ z - z^+ \end{pmatrix} \in \begin{pmatrix} 0 & \hat{A}^T \\ -\hat{A} & 0 \end{pmatrix}\begin{pmatrix} x \\ z^+ \end{pmatrix} + \begin{pmatrix} \partial f(x^+) \\ \partial g^*(z^+) \end{pmatrix}$$

The only difference with applying forward-backward Euler discretization on (2.19) is the z^+ in the first term of the right-hand side, which makes the mixture a little more "implicit" than the forward-backward update. Another option is to use (x^+, z) in that term instead of (x, z^+). Solving for the updates of x^+ and z^+ gives the following algorithm [AH58].

Algorithm 5 (Arrow-Hurwicz)

Initialization *choose any x, z.*

Iteration *repeat*

```
1. $z^+ = \mathbf{prox}_{tg^*}(z + t\hat{A}x)$
2. $x^+ = \mathbf{prox}_{tf}(x - t\hat{A}^T z^+)$
```

Many modifications have been made of this algorithm. In this thesis we studied two relatively recent ones.

3.3.1 Pock-Chambolle

Pock and Chambolle presented a variant of the Arrow-Hurwicz algorithm where they added an extrapolation term $\bar{x}$ with $\theta \in [0, 1]$, and used possibly different step sizes for x and z updates [PCB09, CP11].

Algorithm 6 (Pock-Chambolle)

Initialization *choose any* $\bar{x}$, z.

Iteration *repeat*

```
1. $z^+ = \mathbf{prox}_{\sigma g^*}(z + \sigma\hat{A}\bar{x})$
2. $x^+ = \mathbf{prox}_{\tau f}(x - \tau\hat{A}^T z^+)$
3. $\bar{x} = x^+ + \theta(x^+ - x)$
```

A general convergence $\mathcal{O}(1/k)$ is proven for $\theta = 1$ and $0 < \sigma, \tau < 1/\|\hat{A}\|_2$. For the problems with both f and g being smooth, linear convergence can be reached ($\mathcal{O}(\omega^k)$ with $\omega \in (0, 1)$). No line search has been proposed for this method.

3.3.2 Chen-Teboulle

Similarly, the predictor-corrector proximal method presented by Chen and Teboulle can be interpreted as a variant of the semi-implicit method applied to a reformulation of the primal optimality condition (2.7)

$$0 \in \begin{pmatrix} 0 & 0 & \hat{A}^T \\ 0 & 0 & -I \\ -\hat{A} & I & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} \partial f(x) \\ \partial g(y) \\ 0 \end{pmatrix}$$

which gives

$$\frac{1}{t} \begin{pmatrix} x - x^+ \\ y - y^+ \\ z - z^+ \end{pmatrix} \in \begin{pmatrix} 0 & 0 & \hat{A}^T \\ 0 & 0 & -I \\ -\hat{A} & I & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z^+ \end{pmatrix} + \begin{pmatrix} \partial f(x^+) \\ \partial g(y^+) \\ 0 \end{pmatrix}.$$

Chen and Teboulle's method adds a corrector term as presented in the following algorithm, which they call the predictor-corrector proximal multiplier method.

Algorithm 7 (Chen-Teboulle)

Initialization *choose any* x, y, z.

Iteration *repeat*

$$\begin{array}{rrcl} 1. & \bar{z} & = & z + t(\hat{A}x - y) \\ 2. & x^+ & = & \mathbf{prox}_{tf}(x - t\hat{A}^T\bar{z}) \\ 3. & y^+ & = & \mathbf{prox}_{tg}(y + t\bar{z}) \\ 4. & z^+ & = & z + t(\hat{A}x^+ - y^+) \end{array}$$

Convergence is guaranteed if the optimal solution set is nonempty and the step size satisfies $\varepsilon \leq t \leq \min(\frac{1-\varepsilon}{2}, \frac{1-\varepsilon}{2\|\hat{A}\|_2})$ in each iteration, for some $0 < \varepsilon \leq \min(\frac{1}{3}, \frac{1}{2\|\hat{A}\|_2+1})$. Linear convergence rate is proven with some further assumptions of the problem, such as the existence of a unique optimal point and so on [CT94]. No line search has been proposed for this method.

CHAPTER 4

Numerical Methods Applied on Trace Norm Minimization

In this chapter, we apply the numerical methods described in chapter 3 on our trace norm minimization problem (2.14), with possibility of different combinations of the primal and dual formulation. Interior-point method is included here for the purpose of comparison as a well-developed second-order method.

A few assumptions for the discussion in the chapter: $n \geq \max(p, q)$, $pq \geq n$ and $\hat{A}$ has full column rank, n. For convenience, we also assume $p \geq q$, since $\|Y\|_*$ and $\|Y^T\|_*$ are always identical.

For each method, we show its implementation, linear algebra complexity per iteration, some typical convergence curves, and some observations concerning its performance. Specifically, the two convergence plots shown for each implementation are the minimizations of two randomly generated problem instances of the form 2.14, one of which with size parameters $n = 50,\ p = q = 40,\ \mathbf{rank}(\mathbf{mat}(\hat{A}x) - B) = 10$, the other one with $n = 100,\ p = 80,\ q = 60,\ \mathbf{rank}(\mathbf{mat}(\hat{A}x) - B) = 30$. The relative error $(p - p^*)/p^*$ in the convergence curves is computed with an optimal value as known from the problem generation procedure presented in section 5.1.1. Except for the interior-point method, all the methods terminate when the relative error becomes less than 10^{-5} or the number of iterations exceeds 10^4.

Generally speaking, first-order methods cost less per iteration, but the number of iterations they take to convergence tends to vary with problem complexity, size, and the step size, initial point chosen.

A lookup table for the full name and the shorthand notation of each method is summarized in Table 4.1.

Table 4.1: Lookup table for first-order methods

Notation	Full name	Section
`ADMM`	Alternating direction method of multipliers	3.1.1, 4.2
`DRpd`	Douglas-Rachford splitting on primal-dual	3.1, 4.3
`FIST`	FISTA on dual	3.2.1, 4.4
`TsFB`	Tseng's modified forward-backward splitting on primal-dual	3.2.2, 4.5
`Pock`	Pock-Chambolle	3.3.1, 4.6
`Chen`	Chen-Teboulle on dual	3.3.2, 4.7

4.1 Interior-Point method for Semidefinite Programming

The trace norm minimization problem can be formulated as a semidefinite program (SDP), and the most popular method for solving an SDP is the primal-dual interior-point method. It has the advantage of giving solutions of high accuracy, having robust behavior, and a fast convergence that typically takes some tens of iterations independent of the problem size. However, both the complexity of computation and the memory requirement grow really fast with the problem size.

According to [FHB01, VB96], we can formulate the quadratic regularized trace norm minimization problem (2.10) as a semidefinite program,

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}(t + \mathbf{Tr}(U) + \mathbf{Tr}(V)) \\ \text{subject to} & \begin{pmatrix} I & P^{\frac{1}{2}}x \\ (P^{\frac{1}{2}}x)^T & t \end{pmatrix} \succeq 0 \\ & \begin{pmatrix} U & A(x) - B \\ (A(x) - B)^T & V \end{pmatrix} \succeq 0, \end{array} \tag{4.1}$$

where $t \in \mathbf{R}, U \in \mathbf{S}^p, V \in \mathbf{S}^q$ are optimization variables introduced in addition to the original one, x. The large auxiliary matrix variables introduced by the reformulation thus make it expensive to solve by general-purpose interior-point solvers, where the complexity grows at least as fast as $\mathcal{O}(n^6)$, assuming $p = \mathcal{O}(n)$ and $q = \mathcal{O}(n)$. Liu and Vandenberghe have developed a more efficient method by exploiting the structure of the semidefinite program reformulation, and the cost per iteration is reduced to $\mathcal{O}(n^4)$, which is comparable to the cost of solving the approximation problem in the Frobenius norm, that is, a least squares problem of size $pq \times n$ [LV09].

As noted in the literature, it takes only a small number of iterations for the interior-point method to reach a desirable accuracy, and there is a steady progress in each iteration.

4.2 ADMM

Applying ADMM to our trace norm minimization problem, we obtain the following implementation.

Implementation 1 (`ADMM`)

Initialization *choose any* y $(= \mathbf{vec}(B))$ *and* z $(= 0)$.

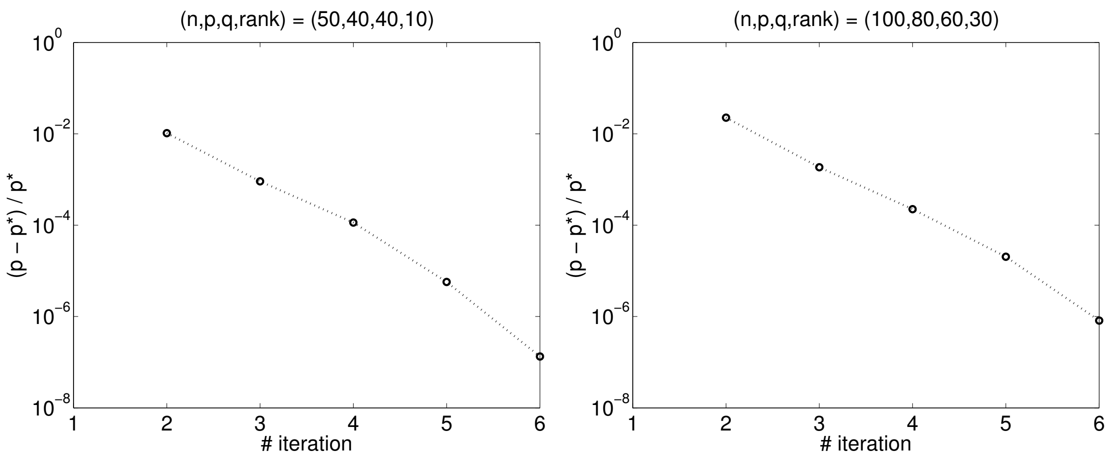


Figure 4.1: Interior-point method convergence curve

The missing data point is due to a primal objective that is smaller than the optimum because of the allowance of infeasible points in the primal-dual interior-point method.

Iteration *repeat*

$$\begin{array}{rrcl} 1. & x^+ & = & (P + t\hat{A}^T\hat{A})^{-1}\hat{A}^T(ty - z) \\ 2. & U\mathbf{diag}(s)V^T & = & \mathbf{mat}(\hat{A}x + \frac{1}{t}z) - B \\ & y^+ & = & \mathbf{vec}(B + U\mathbf{diag}(\max(0, s - \frac{\mathbf{1}}{t}))V^T) \\ 3. & z^+ & = & z + t(\hat{A}x^+ - y^+) \end{array}$$

Again, $U\mathbf{diag}(s)V^T$ in step 2 refers to the compact singular value decomposition of the right hand side, which is the most costly step in each iteration, taking $\mathcal{O}(p^2q)$ flops. Solving the linear equation $(P+t\hat{A}^T\hat{A})x^+ = \hat{A}^T(ty-z)$ in step 1 can be even more expensive if it were implemented naively. However, some preprocessing can significantly reduce the complexity. For example, if t is fixed, only one Cholesky factorization for $P+t\hat{A}^T\hat{A}$ is needed before the iterations start, and then computing $\hat{A}^T(ty - z)$ and using forward and backward substitution to solve for x^+ cost $\mathcal{O}(npq)$ and $\mathcal{O}(n^2)$, respectively. However, the step size t is usually made adaptable in practice to improve convergence. In this case, we use simultaneous diagonalization as described in [LHV13] for a general P, where the complexity for solving the linear equation remains $\mathcal{O}(n^2)$. First take a Cholesky factorization,

$$LL^T = P + \hat{A}^T\hat{A},$$

and then an eigenvalue decomposition,

$$Q\Lambda Q^T = L^{-1}PL^{-T}.$$

Noting therefore that

$$\begin{aligned} P &= LQ\Lambda Q^T L^T \\ \hat{A}^T\hat{A} &= LQ(I-\Lambda)Q^T L^T, \end{aligned}$$

and $(P + t\hat{A}^T\hat{A})x = b$ can be solved by

$$x = L^{-T}Q(tI + (1-t)\Lambda)^{-1}Q^T L^{-1}b.$$

If desired, the Cholesky factorization L can be computed by taking a QR factorization, that is,

$$\tilde{Q}L^T = \begin{pmatrix} P^{\frac{1}{2}} \\ \hat{A} \end{pmatrix},$$

as in the case discussed above. The same technique for solving linear equations is used for other methods discussed in the following sections.

An observation is made concerning the effect of different choices of t such that for each problem size, a certain choice of t usually gives the fastest convergence, while larger or smaller choices give slower ones. This can be explained in light of the simultaneous convergence of the primal and dual residuals defined as

$$\begin{aligned} r_{\mathrm{p}} &= \hat{A}x^+ - y^+ \\ r_{\mathrm{d}} &= t\hat{A}^T(y^+ - y). \end{aligned}$$

Larger t places a larger penalty on violation of primal feasibility, and thus may result in a smaller r_{p}. Smaller t may produce a smaller r_{d} according to its definition, while the resulting r_{p} might be larger. Thus, one intuitive example of step adjusting as mentioned in [BPC11] is

$$t^+ = \begin{cases} \beta t & \|r_\mathrm{p}\|_2 > \mu \|r_\mathrm{d}\|_2 \\ t/\beta & \|r_\mathrm{d}\|_2 > \mu \|r_\mathrm{p}\|_2 \\ t & \text{otherwise} \end{cases}$$

which seeks to keep r_p and r_d within a factor μ of each other. The parameters $\beta, \mu > 1$ are typically chosen as $\beta = 2$ and $\mu = 10$.

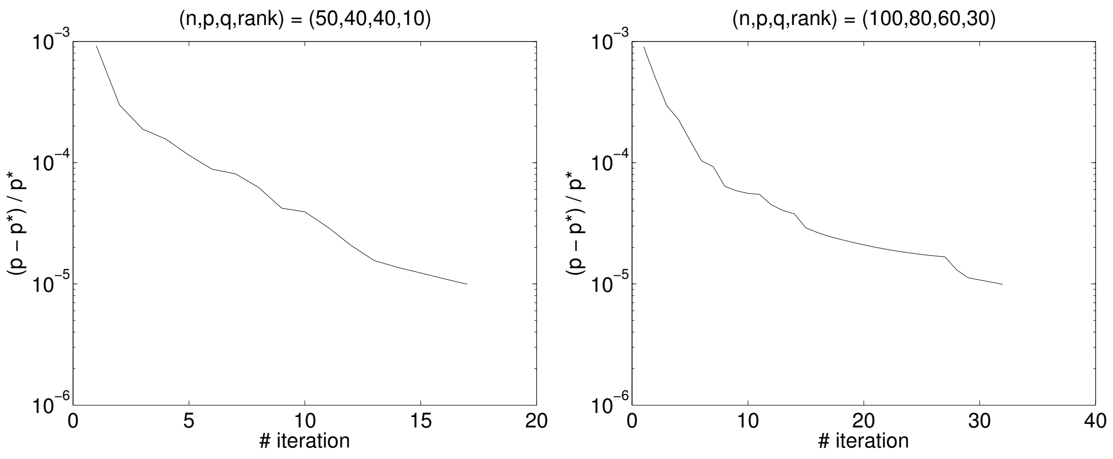


Figure 4.2: ADMM convergence curve

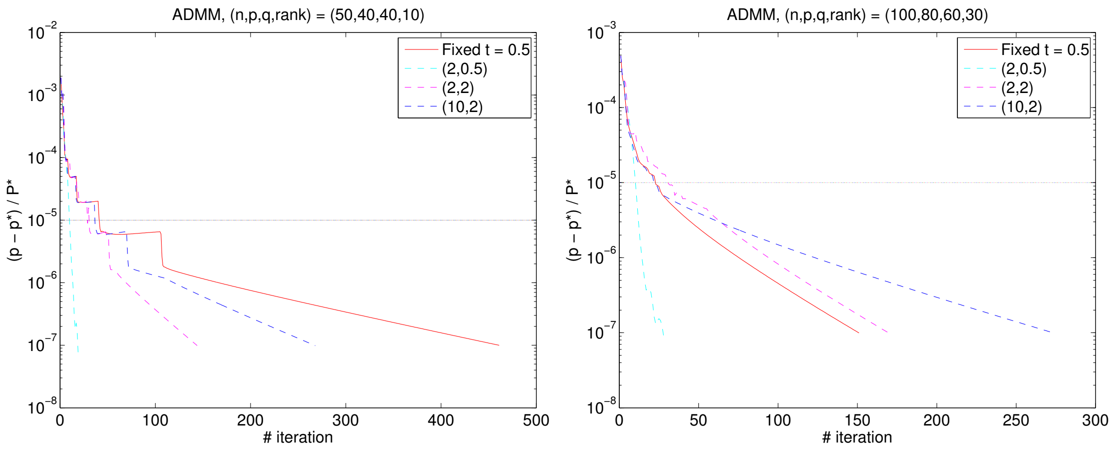


Figure 4.3: ADMM convergence curve. Dashed line curves use adaptive step size with parameters (μ, β).

Lastly, the expression shown in parenthesis for initialization step is the initial point chosen in our implementation, which also affects the convergence. According to [LHV13], y is initialized to zero, but we find initializing it to $\mathbf{vec}(B)$ generally gives faster convergence.

As can be seen in Figure 4.2, ADMM exhibits steady and fast progress in each iteration, as compared to other first-order methods. Moreover, in Figure 4.3, we see that different choices for the parameters of step size adaption give different rate of convergence, and its effect varies with problem sizes. Also, while $\mu = 2,\ \beta = 0.5$ gives a superior performance in these two instances, it does not reach convergence in some other instances.

4.3 Douglas-Rachford Splitting on Primal-Dual

There are several ways to split the zero inclusion problem (2.19). For example, if we take

$$\begin{aligned} F(x,z) &= \begin{pmatrix} 0 & \hat{A}^T \\ -\hat{A} & 0 \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} \\ G(x,z) &= \begin{pmatrix} \partial f(x) \\ \partial g^*(z) \end{pmatrix} \end{aligned} \tag{4.2}$$

for (**Alg. 1**) we obtain the following implementation.

Implementation 2 (`DR1A`)

Initialization *choose any* $\tilde{x}$ $(= 0)$ *and* $\tilde{z}$ $(= 0)$.

Iteration *repeat*

$$\begin{aligned}
1.\quad x^+ &= (I + tP)^{-1}\tilde{x} \\
z^+ &= P_C(\tilde{z} - t\mathbf{vec}(B)) \\
2.\quad \bar{x}^+ &= (I + t^2\hat{A}^T\hat{A})^{-1}(2x^+ - \tilde{x} - t\hat{A}^T(2z^+ - \tilde{z})) \\
\bar{z}^+ &= 2z^+ - \tilde{z} + t\hat{A}\bar{x}^+ \\
3.\quad \tilde{x}^+ &= \tilde{x} + \bar{x}^+ - x^+ \\
\tilde{z}^+ &= \tilde{z} + \bar{z}^+ - z^+
\end{aligned}$$

Recall from chapter 2 that P_C is the Euclidean projection onto set C, as defined in (2.15). The z update in the first step is done by taking an SVD of the matrix form of the parameter of P_C, and then truncating to one those singular values larger than one. Specifically, it is done in two steps,

$$\begin{aligned}
U\mathbf{diag}(s)V^T &= \mathbf{mat}(\tilde{z} - t\mathbf{vec}(B)) \\
z^+ &= \mathbf{vec}(U\mathbf{diag}(\min(1, s))V^T),
\end{aligned}$$

where $U\mathbf{diag}(s)V^T$ refers to the compact SVD of the right hand side. Same rule applies to the following sections.

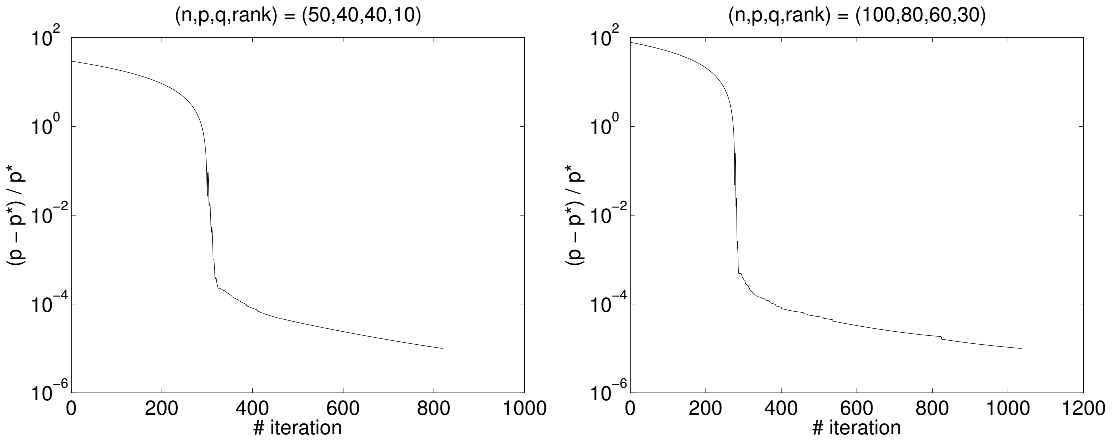


Figure 4.4: DR1A convergence curve

Interchanging the operator F and G in the previous case, we obtain a different implementation. We can also take another splitting such as

$$\begin{aligned} F(x,z) &= \begin{pmatrix} 0 & \hat{A}^T \\ -\hat{A} & 0 \end{pmatrix} \begin{pmatrix} x \\ z \end{pmatrix} + \begin{pmatrix} \partial f(x) \\ 0 \end{pmatrix} \\ G(x,z) &= \begin{pmatrix} 0 \\ \partial g^*(z) \end{pmatrix}. \end{aligned}$$

Again, interchanging F and G in the previous case gives us another implementation. However, the performance of all four combinations appears to be almost identical, only off by one iteration. Thus, we refer to these four implementations as `DRpd`. The curve for `DRpd` exhibits a slow convergence for the most part and a fast drop at one point, which varies with the step size chosen. A scheme for adaptive step size similar to that for ADMM might improve these implementations.

Now consider the complexity of these methods. Each of them does two matrix-vector multiplications of $\mathcal{O}(npq)$ and one SVD that costs $\mathcal{O}(p^2q)$ in one iteration.

4.4 FISTA on Dual

Since applying FISTA on the primal problem (2.14) gives a problem as difficult to solve in each iteration as the problem itself, in this thesis we apply FISTA on the dual problem (2.16). We use the dual variable z, take f in (**Alg. 3**) to be $\frac{1}{2}z^T\hat{A}P^\dagger\hat{A}^Tz$ and g to be $\mathbf{vec}(B)^Tz + I_C(z)$, then obtain the following implementation.

Implementation 3 (`FIST`)

Initialization *choose any* $z = z^-$ $(= 0)$.

Iteration *for* $k \geq 1$, *repeat*

$$\begin{aligned} 1.\quad \bar{z} &= z + \tfrac{k-2}{k+1}(z - z^-) \\ 2.\quad z^+ &= P_C(\bar{z} - t(\mathbf{vec}(B) + \hat{A}P^\dagger\hat{A}^T\bar{z})) \end{aligned}$$

For a general dense P, we can take its Cholesky factorization during preprocessing, so the complexity for each iteration becomes one SVD and two $\mathcal{O}(npq)$ multiplications.

The rate of convergence of `FIST` slows down really soon, and the objective value even jumps up after the initial convergence.

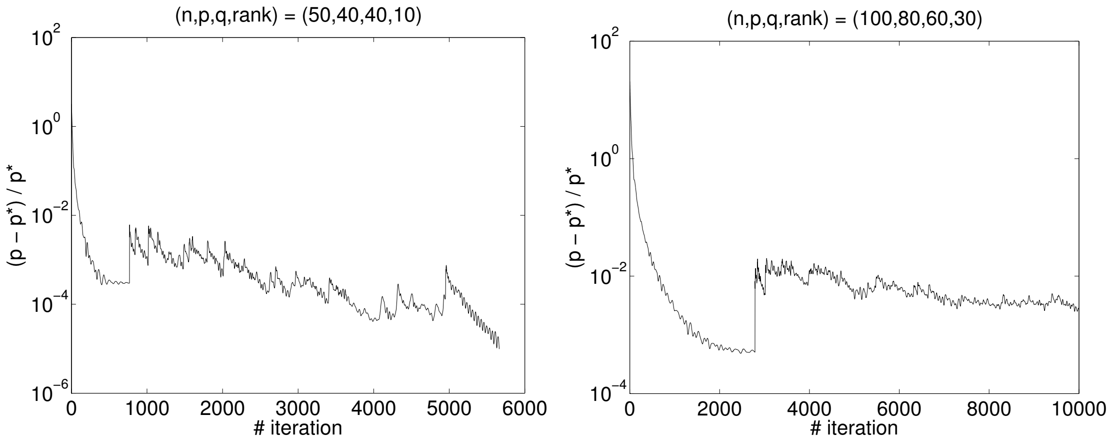


Figure 4.5: `FIST` convergence curve

4.5 Tseng's Modified Forward-Backward Splitting on Primal-Dual

Apply (**Alg. 4**) on the primal-dual optimality condition (2.19) with $F(x, z)$ and $G(x, z)$ split shown in (4.2). Thus, F is single-valued and Lipschitz continuous with $L = \|\hat{A}\|_2$.

Implementation 4 (`TsFB`)

Initialization *choose any* (x, z) $(= (0, 0)) \in \mathbf{dom} A$.

Iteration *repeat*

$$\begin{aligned} 1.\quad \bar{x} &= (I + tP)^{-1}(x - t\hat{A}^T z) \\ \bar{z} &= P_C(z + t(\hat{A}x - \mathbf{vec}(B))) \\ 2.\quad x^+ &= \bar{x} - t\hat{A}^T(\bar{z} - z) \\ z^+ &= \bar{z} + t\hat{A}(\bar{x} - x) \end{aligned}$$

In each iteration, one SVD and four $\mathcal{O}(npq)$ matrix-vector multiplications are performed.

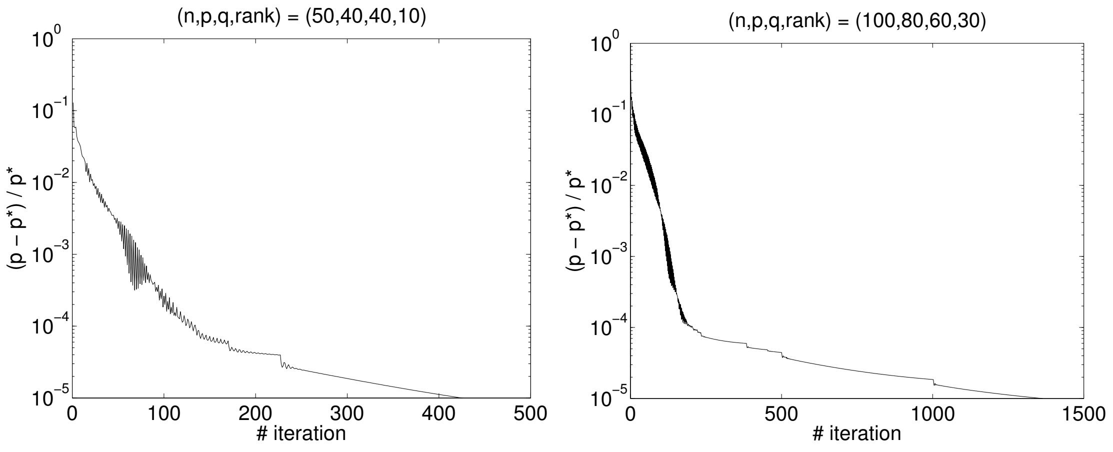


Figure 4.6: `TsFB` convergence curve

Generally speaking, `TsFB` exhibits two phases in the convergence curve, first of which converges faster and has interesting oscillating behavior, second of which slower.

4.6 Pock-Chambolle

We use the same step size t for both primal and dual updates.

Implementation 5 (`Pock`)

Initialization *choose any* $\bar{x}$ $(= 0)$*,* z $(= 0)$.

Iteration *repeat*

$$\begin{array}{rrcl} 1. & z^+ & = & P_C(z + t(\hat{A}\bar{x} - \mathbf{vec}(B))) \\ 2. & x^+ & = & (I + tP)^{-1}(x - t\hat{A}^T z^+) \\ 3. & \bar{x} & = & x^+ + \theta(x^+ - x) \end{array}$$

In each iteration, one SVD and two $\mathcal{O}(npq)$ matrix-vector multiplications are performed.

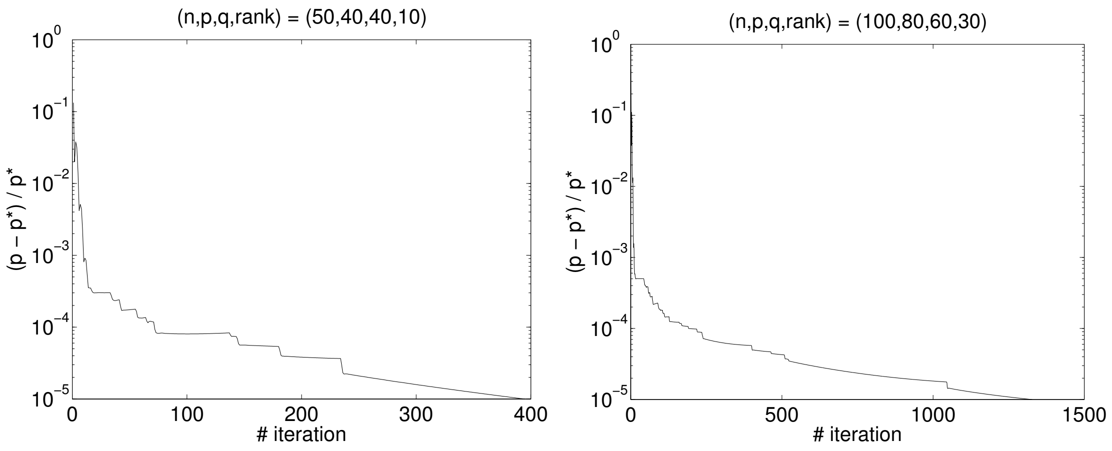


Figure 4.7: `Pock` convergence curve

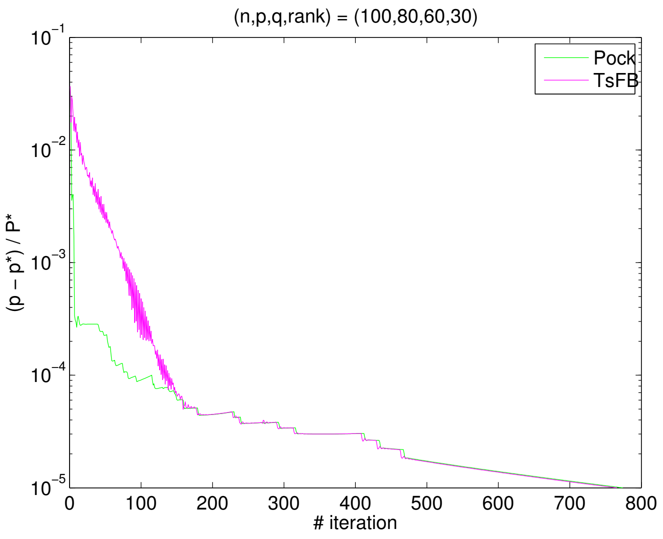


Figure 4.8: Comparison of `TsFB` and `Pock`

The convergence of `Pock` is also roughly a two-phase one. It exhibits a fast convergence to a modest accuracy and then a rather slow one thereafter. A comparison is made between `TsFB` and `Pock` shown in Figure 4.8. Although `Pock` reaches modest accuracy a lot sooner, both methods typically converge to higher accuracy with almost the same pace.

4.7 Chen-Teboulle on Dual

Applying Chen and Teboulle's method to the dual optimality condition (2.18) gives us $f(z) = \mathbf{vec}(B)^T z + I_C(z),\ g(w) = \frac{1}{2} w^T P^\dagger w$ and $\hat{A}^T z = w$ for (**Alg. 7**), and we have the following implementation.

Implementation 6 (`Chen`)

> ***Initialization*** *choose any* $z\ (=0), w\ (=0), x\ (=0)$.
>
> ***Iteration*** *repeat*

$$
\begin{array}{rrcl}
1. & \bar{x} & = & x + t(\hat{A}^T z - w) \\
2. & z^+ & = & P_C(z - t(\hat{A}\bar{x} + \mathbf{vec}(B))) \\
3. & w^+ & = & (I + tP^\dagger)^{-1}(w + t\bar{x}) \\
4. & x^+ & = & x + t(\hat{A}^T z^+ - w^+)
\end{array}
$$

In each iteration, one SVD and three $\mathcal{O}(npq)$ matrix-vector multiplications are performed.

The convergence as shown in Figure 4.9 is steady but rather slow even for small problems. However, we can see that the corrector step added from the pure semi-implicit update has a nice effect on the convergence curve, as compared to Figure 4.10.

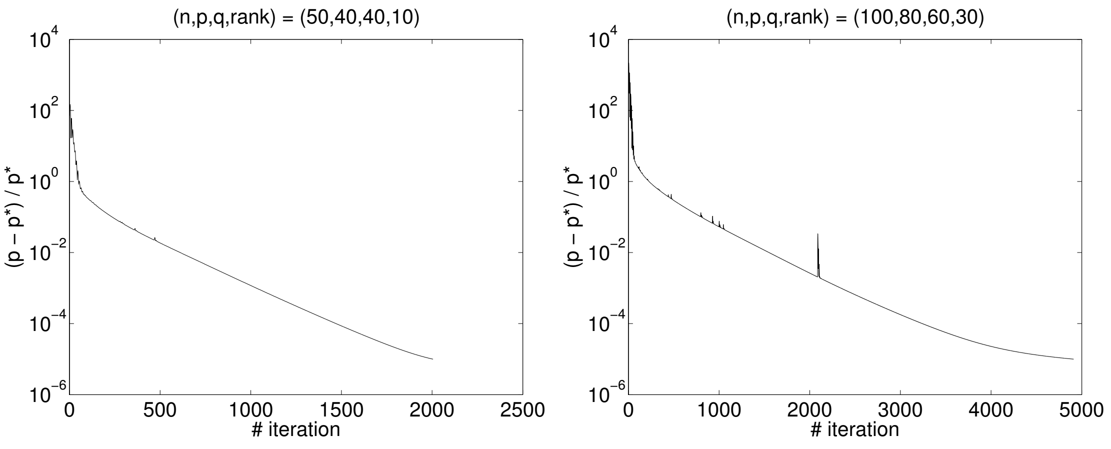


Figure 4.9: `Chen` convergence curve

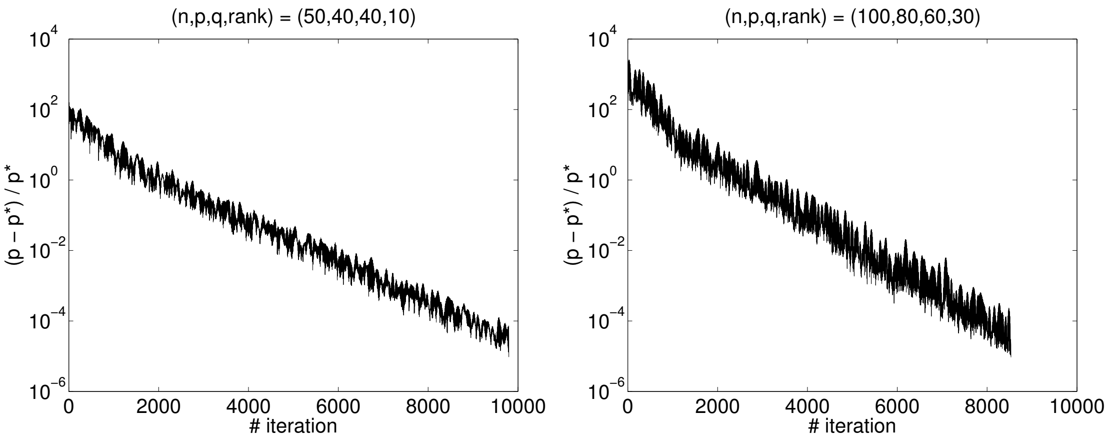


Figure 4.10: Convergence curve for pure semi-implicit method, implemented as `Chen` but without step 4, the corrector step.

4.8 Summary

Considering all the first-order splitting methods discussed above, we see that FISTA on dual (`FIST`) and Chen-Teboulle on dual (`Chen`) do not converge to acceptable accuracy within reasonable number of iterations; Douglas-Rachford splitting on primal-dual (`DRpd`), Tseng's modified forward-backward splitting on primal-dual (`TsFB`), and Pock-Chambolle (`Pock`) converge to modest accuracy within reasonable number of iterations, while `Pock` gives faster initial convergence;

only `ADMM` reaches high accuracy and demonstrates fast initial convergence. Their behaviors can be summarized in Figure 4.11.

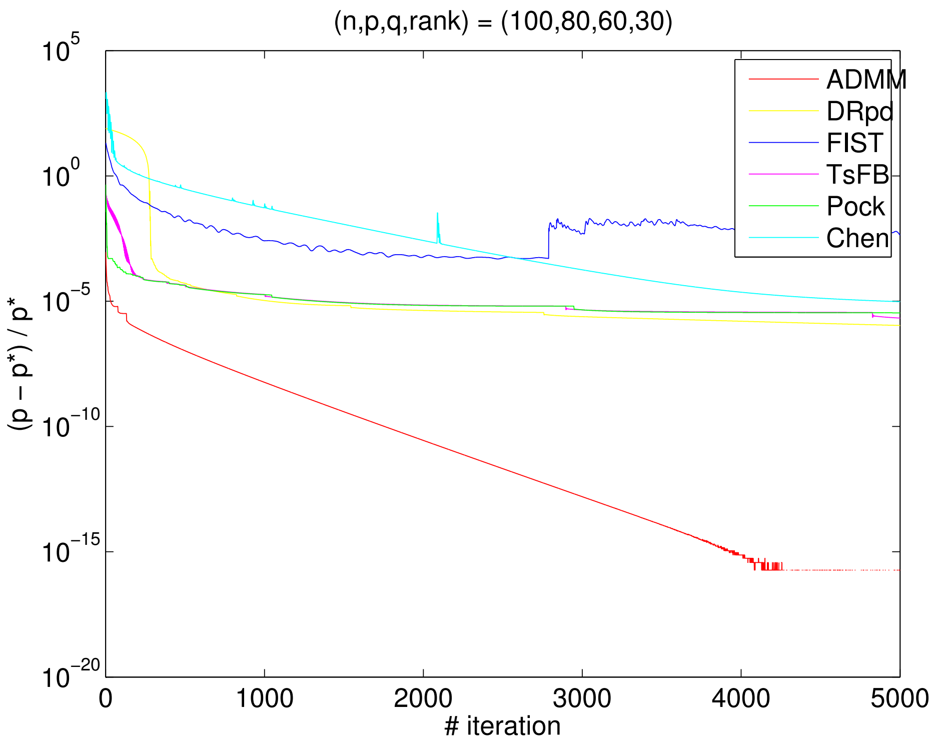
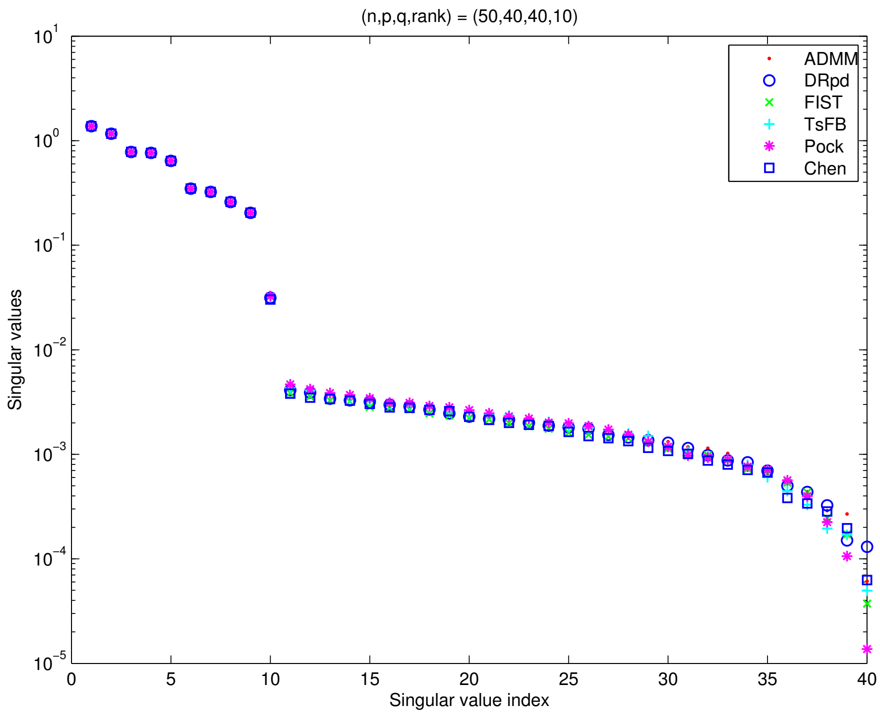


Figure 4.11: Comparison of convergence curves

We can also observe that the solutions that these methods find to the quadratically regularized trace norm minimization do tend to be low-rank. Specifically, if we take only the singular values within two orders of the largest one, then the optimal rank is recovered. The range of the corresponding left singular vectors also converges to that of the optimal solution, which is verified by taking the angle between the optimal and the recovered subspaces. For the two instances shown in this chapter, the angles are $4.49\,^\circ$ and $8.65\,^\circ$ respectively.



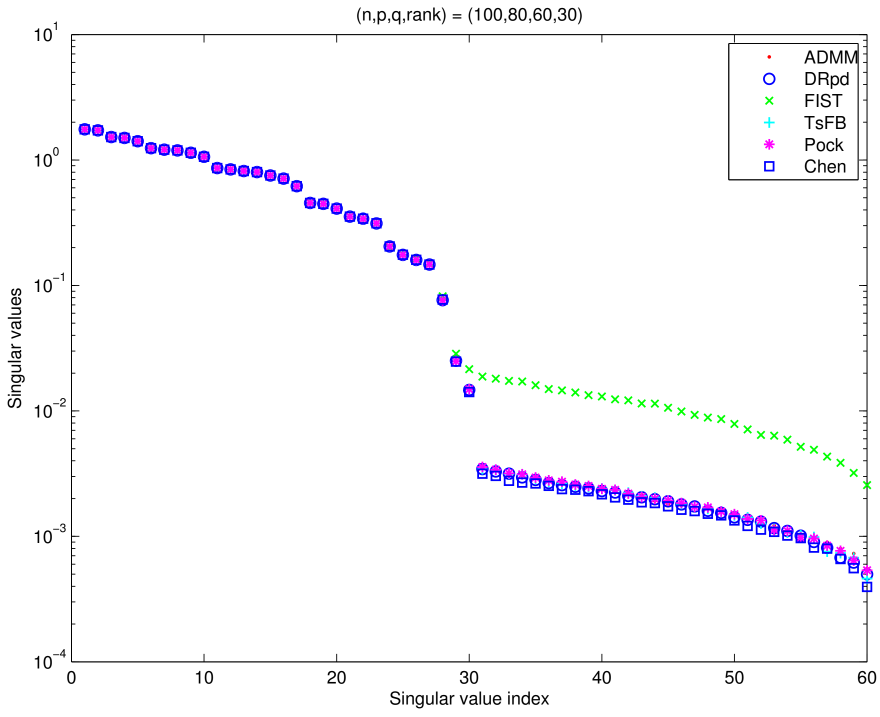


Figure 4.12: Singular values of solutions

In the bottom figure `FIST` does not reach the same accuracy because it reached the iteration limit before converging to the desired value.

CHAPTER 5

Experimental Results

In this chapter we give numerical experiments that compare the performance of each method over different problem sizes.

All the experiments in this thesis are run on a 2.83 GHz quad-core Intel Q9550 processor with 8 GB of memory. The primal-dual interior-point method is found in CVXOPT (version 1.1.5) written in Python (version 2.7.3) [GCC4.6.3] [LV09], and all the other implementations use MATLAB 7.13.0.564 (R2011b).

5.1 Randomly Generated Data

In this section we first explain how the random problems used to test the methods are generated, and then we show extensive experiments of all the first-order splitting methods on some medium-sized problems, which then become the basis for selecting promising methods for large-scale problems.

5.1.1 Random Problem Generation

We are not certain if all randomly generated problems will have optimal solutions. Hence we want to generate problems that are known to have optimal solutions. This method of problem generation also allows us to generate a problem with specified rank of the matrix $\mathbf{mat}(\hat{A}x) - B$, which is an essential parameter with applications on system identification.

The parameters that define the problem are $P, \hat{A}, B$, the optimization variable is x, and the supplemental variables are y and z. An optimal solution along with the associated problem parameters should satisfy the generalized KKT conditions [BV04], which is another way to describe optimality conditions. Consider a reformulation of (2.14),

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Px + \|\mathbf{mat}(y)\|_* \\ \text{subject to} & \hat{A}x - \mathbf{vec}(B) = y, \end{array}$$

with z being the multiplier of the equality constraint. The KKT conditions are

$$\begin{aligned} &\hat{A}x - \mathbf{vec}(B) = y \\ &Px + \hat{A}^T z = 0 \\ &\|\mathbf{mat}(y)\|_* - z^T y = 0 \\ &\|\mathbf{mat}(z)\|_2 \leq 1. \end{aligned}$$

Moreover, for a convex optimization problem, points that satisfy the KKT conditions are also guaranteed to be optimal. Thus, we can randomly generate some of the parameters and solutions, and use the equations given by the KKT conditions to derive the rest, as proposed in the following implementation.

Implementation 7 (Random problem generation)

1. *Randomly generate* $P \succeq 0,\ \hat{A},\ \|\mathbf{mat}(z^*)\|_2 \leq 1$

2. *Solve for a* y *that satisfies* $\|\mathbf{mat}(y)\|_* - z^{*T} y = 0$

3. *Solve for* x^* *according to* $Px^* + \hat{A}^T z^* = 0$

4. *Solve for* B *according to* $\hat{A}x^* - \mathbf{vec}(B) = y$.

The random matrices are generated by the `randn` function in MATLAB. The matrix P can be chosen to be dense, diagonal, or a multiple of the identity matrix. The matrix $\mathbf{mat}(z^*)$ is generated randomly, and its singular values are scaled so that the m^{th} singular value becomes one, and then all those larger than one are truncated to one. In step 2, we first randomly generate the first m singular values for $\mathbf{mat}(y)$, set the rest to zero, and multiply it with the corresponding m left and right singular vectors of $\mathbf{mat}(z^*)$. Specifically, if USV^T is a singular value decomposition of $\mathbf{mat}(z^*)$ and $\tilde{Y}$ is the matrix whose first m diagonal elements are randomly generated positive values, then $y = \mathbf{vec}(U\tilde{Y}V^T)$.

5.1.2 Method Selection

Experiments in this section are conducted over problems with around a hundred variables. Again, we assume that $n \geq \max(p, q)$, $pq \geq n$ and $\hat{A}$ has full column rank, n, which hold generally in applications.

Table 5.1: Performance of Douglas-Rachford splitting type methods over randomly generated problems of medium sizes. Under each method, the left column is the number of iteration, the right column is the cpu time in seconds.

n	p	q	rank	ADMM(0)		ADMM		DRpd	
50	40	40	30	65.2	0.33	14.8	0.08	1414.0	12.10
			10	63.3	0.28	12.5	0.06	3481.7	27.46
	20	20	10	52.4	0.12	16.4	0.04	1756.6	6.41
	40	20	10	58.5	0.15	14.4	0.03	4264.6	19.49
100	80	80	60	120.9	3.35	13.7	0.37	6203.5	336.83
			30	113.5	3.07	11.5	0.29	7535.6	410.75
			10	108.4	3.07	8.9	0.24	8969.1	518.80
	40	40	30	92.3	0.51	15.6	0.09	8806.6	101.34
			10	87.2	0.48	11.4	0.07	8214.2	75.84
	20	20	10	86.5	0.19	14.0	0.03	1184.7	5.00
	80	60	30	108.8	1.88	10.5	0.19	11957.8*	453.99
			10	110.2	2.03	8.7	0.15	7070.7*	268.76
	40	30	10	89.4	0.34	11.7	0.05	6226.0*	44.74
	80	40	10	104.2	1.14	10.4	0.12	5635.0	137.89
	80	20	10	105.8	0.34	10.6	0.04	9233.5*	67.47
	40	20	10	81.6	0.22	12.3	0.04	1828.6	9.75

*Some instance(s) reached maximal iteration (40000) without converging to an acceptable solution.

Table 5.1, 5.2, and 5.3 shows, for each method, the average number of iterations and cpu time in seconds over 10 randomly generated problems, with 16 different problem dimensions. The methods terminate when the relative error is less the 10^{-5}. The matrix P is taken to be diagonal in all cases. A couple observations are drawn from the experiments.

Here `ADMM`(0) uses step size 1.0 and initial point $y = 0$, ADMM uses step size 0.5 and initial point $y = \mathbf{vec}(B)$, and `DRpd` uses 1.0 step size. We see that changing step size and initial point for `ADMM` improves its speed to convergence by an order. Using a smaller step size for `DRpd` also improves its speed from that shown in Table 5.1. Roughly speaking, we see that `ADMM` takes longer time as p, q and rank grow larger, and `DRpd` takes longer with growing dimensions.

Table 5.2: Performance of forward-backward splitting methods over randomly generated problems of medium sizes. Under each method, the left column is the number of iteration, the right column is the cpu time in seconds.

n	p	q	rank	FIST		TsFB	
50	40	40	30	2929.7	22.89	210.4	1.76
			10	9290.9*	52.07	229.2	1.99
	20	20	10	3088.4	5.90	194.3	0.88
	40	20	10	5748.3	13.36	213.0	0.96
100	80	80	60	13324*	1211.7	253.0	14.48
			30	18521*	1300.2	265.6	15.84
			10	18124*	1311.5	384.0	23.22
	40	40	30	10687	61.19	176.6	2.44
			10	12377*	68.53	207.5	2.73
	20	20	10	2198.8	4.66	147.3	0.72
	80	60	30	13272	551.44	228.6	9.65
			10	15324	530.10	308.6	12.20
	40	30	10	7754.2	29.14	175.8	1.38
	80	40	10	10240	186.57	259.2	6.88
	80	20	10	11321*	39.67	179.5	1.45
	40	20	10	5081.3	12.96	161.3	0.94

*Some instance(s) reached maximal iteration (40000) without converging to an acceptable solution.

In Table 5.2, we see that `TsFB` takes shorter time to convergence when the rank is fuller. While `FIST` exhibits similar trend, it is less obvious. The number of iterations for `TsFB` sometimes varies a lot due to the two-phase feature of its convergence curve.

Table 5.3: Performance of semi-implicit splitting methods over randomly generated problems of medium sizes. Under each method, the left column is the number of iteration, the right column is the cpu time in seconds.

n	p	q	rank	Pock		Semi		Chen	
50	40	40	30	205.5	1.00	2024.5	9.96	1261.8	6.15
50	40	40	10	226.1	0.92	6592.8	30.71	1592.9	7.61
50	20	20	10	192.4	0.48	1753.1	3.33	875.5	1.69
50	40	20	10	202.4	0.41	3125.8	6.80	1062.5	2.42
100	80	80	60	245.1	5.33	4210.7	102.37	2615.5	58.96
100	80	80	30	245.2	5.38	8346.3	194.13	3106.9	67.53
100	80	80	10	209.7	4.58	24524	572.18	3569.8	76.50
100	40	40	30	169.9	0.89	2140.2	11.50	1312.2	7.17
100	40	40	10	191.6	0.97	7355.6	38.82	1669.2	8.92
100	20	20	10	148.9	0.31	1729.3	4.02	885.9	2.10
100	80	60	30	194.8	2.79	5856.4	86.57	2537.0	37.59
100	80	60	10	211.2	2.62	19947	274.60	2759.9	39.38
100	40	30	10	171.1	0.61	4766.9	17.88	1441.1	5.44
100	80	40	10	219.5	1.94	12561.3	115.04	2528.4	22.36
100	80	20	10	169.5	0.48	5688.3	17.36	1575.2	4.95
100	40	20	10	162.3	0.40	2933.7	7.89	1132.4	3.06

In Table 5.3, we see that the average number of iterations of `Pock` does not vary much according to problem dimensions, while the time it takes does. Compared to pure semi-implicit implementation, `Chen` has consistently better performance. Both of them obviously take longer time as p, q grow larger, but also as the rank grows smaller.

In general, listing the methods in the order of decreasing performance, we have `ADMM`, `Pock`, `TsFB`, `DRpd`, `Chen`, and `FIST`. We then choose `ADMM`, `TsFB`, and `Pock` to proceed to large-scale problems.

5.1.3 Large-Scale Data

Experiments on 12 sets of problems with different dimensions listed in Table 5.4 are conducted.

Table 5.4: Sizes of problems for large-scale experiments

Problem set #	n	p	q	rank
1	50	20	10	3
2	250	200	100	30
3	500	250	50	10
4	500	250	50	30
5	500	200	100	30
6	500	450	50	10
7	1000	200	100	30
8	1000	940	30	10
9	500	400	400	30
10	500	400	400	100
11	1000	800	100	10

The performance is compared in Figure 5.1, where the order of the problem sets is arranged roughly according to the cpu time it takes interior-point solver so solve the problem. A version of `ADMM` with adaptive step size with $\mu = 10, \beta = 2$ is included in the comparison. We stop the first-order methods when the accuracy reaches 10^{-5}, and the interior-point solver terminates with that better than 10^{-6}.

Each entry is the average result of running on 10 different problems, except for the entry for the interior-point method, which is generated with one instance. The data is considered reliable since the performance of the interior-point method is stable over different problem instances of the same dimensions, while that of first-order methods may vary.

We see that for larger problems, `ADMM`, like interior-point method, uses a relatively small number of iterations to reach convergence. In the bottom figure, interior-point method takes longer time than `ADMM` on large problems and even

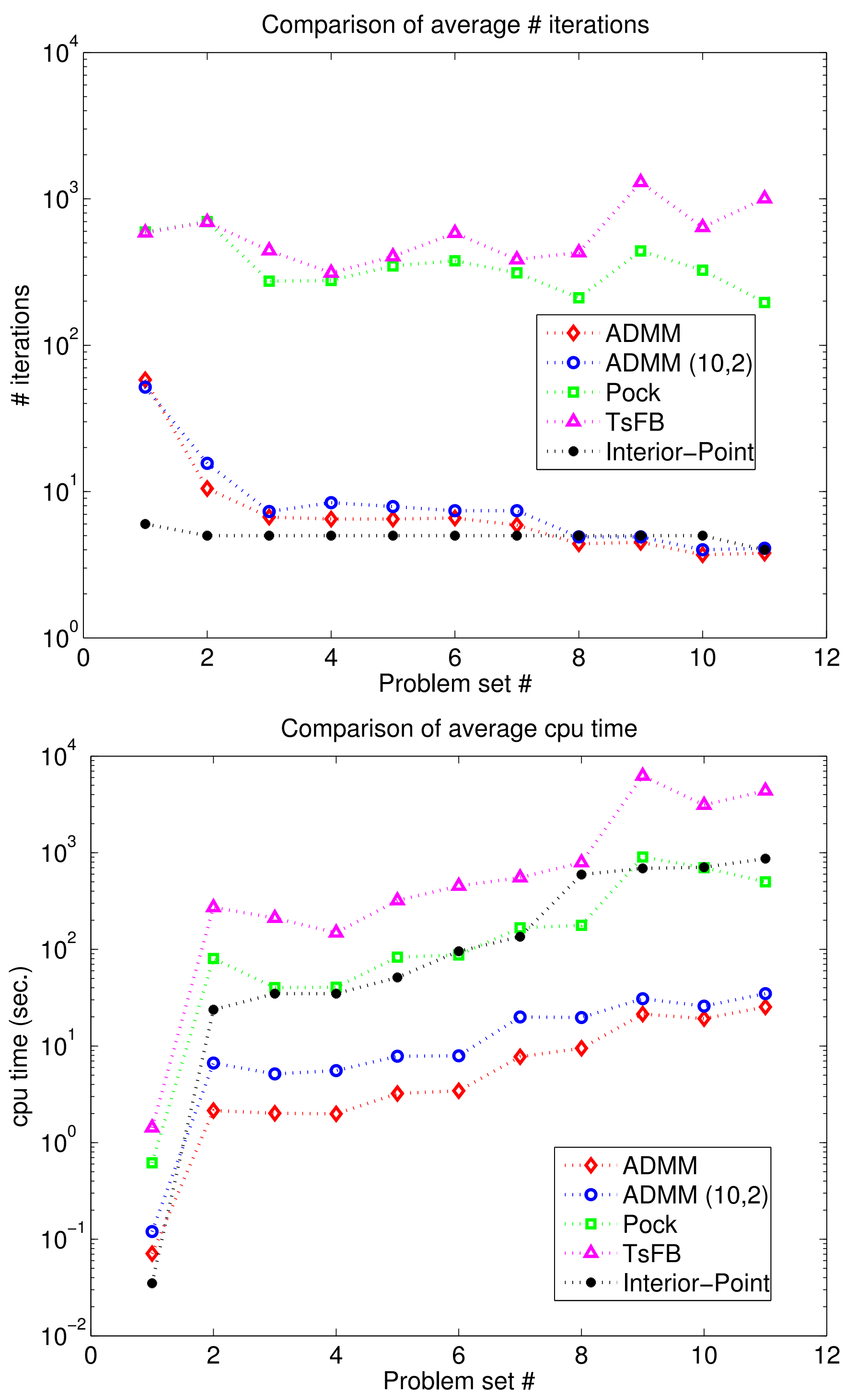


Figure 5.1: Performance comparison over large-scale problems

`Pock` is faster in some problems, especially those whose $p \gg q$. This result justifies the use of first-order methods over interior-point methods on large-scale problems.

5.2 System Identification Data

Finally, we test the quality of trace norm minimization as a heuristic for rank minimization on real-world data collected by [DDD94].

Following the implementation of CVXOPT [LV09], we choose $r = \min(\frac{30}{k}, \frac{N+2}{k+l+1} + 1)$. It can be noticed that r is chosen so that the resulting $YU^{\perp}$ is rather rectangular. Specifically, since the order of the systems in practice rarely exceeds 30, we choose the column dimension kr to be less than or equal 30. It is better small in the sense that the estimated column space may be more accurate, and that it requires a smaller number of variables, which is desirable both for the interior-point methods and the first-order methods.

We first fit the problem (1.6) to the general form (2.14) discussed in the thesis, with optimization variables $\tilde{y}(t) = y(t) - y_{\text{meas}}(t), t = 0, \ldots, N$. Consider the single-input-single-output case, we take $P = 2\gamma I$, $B = -\mathbf{mat}(\hat{A}y_{\text{meas}})$, and

$$\hat{A} = \mathbf{mat}(\mathbf{vec} \begin{pmatrix} u_1^{\perp} & u_2^{\perp} & \cdots & u_r^{\perp} & \cdots & u_{N-r+1}^{\perp} & \cdots & 0 & 0 \\ 0 & u_1^{\perp} & \cdots & u_{r-1}^{\perp} & \cdots & u_{N-r}^{\perp} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & u_1^{\perp} & \cdots & u_{N-2r+2}^{\perp} & \cdots & u_{N-r}^{\perp} & u_{N-r+1}^{\perp} \end{pmatrix}),$$

where $u_i^{\perp}$, $i = 1, \ldots, N-r+1$ are the row vectors of $U^{\perp}$.

We use $N = 198$ in the following experiments. Shown in Figure 5.2 is one example of the effect of using different γ, where a clear-cut in the singular values can be identified when γ gets smaller while maintaining an acceptable identification error. In some other cases, for example, as shown in Figure 5.3, the solutions to the trace norm minimization fail to suggest an obvious system order.

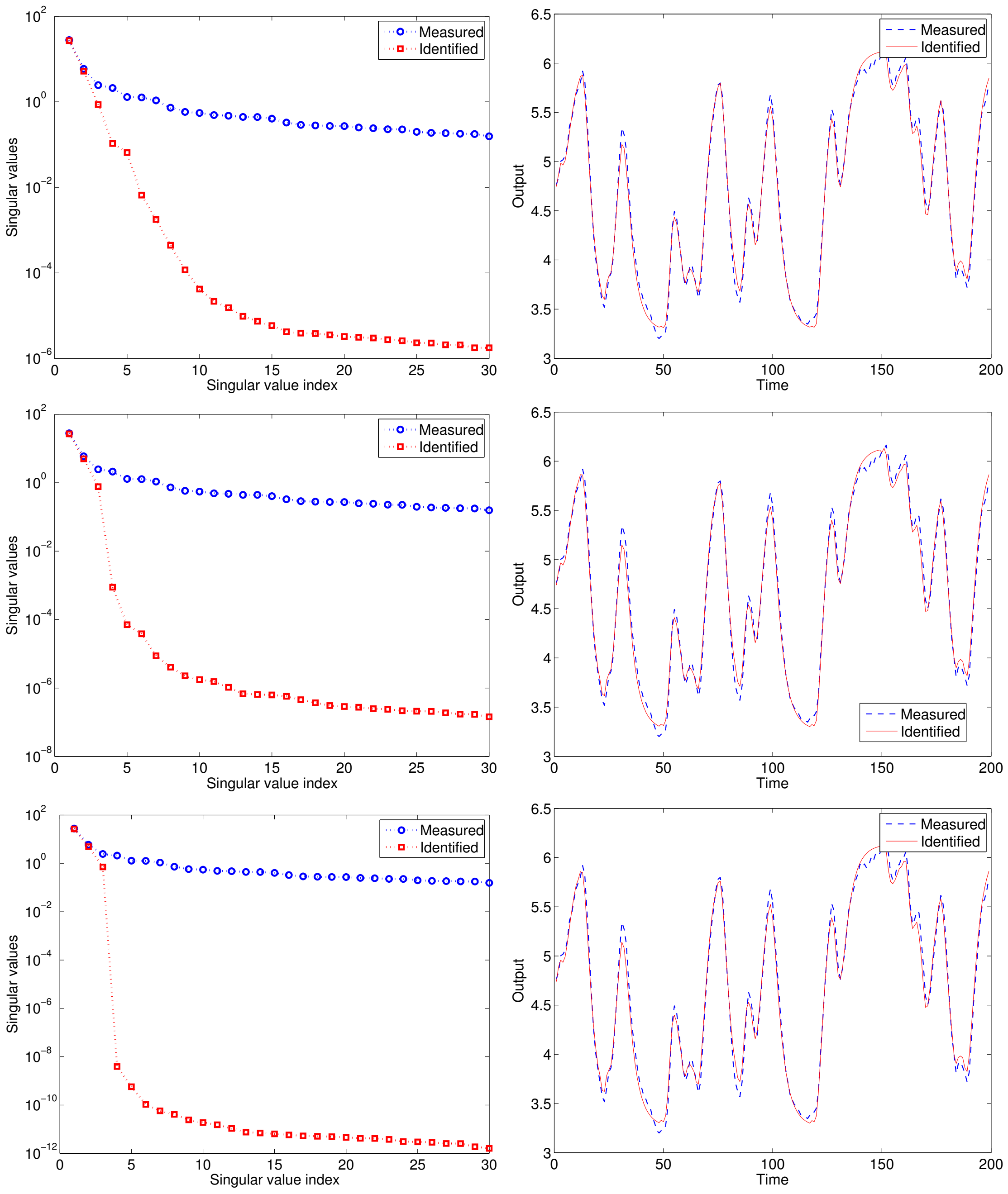


Figure 5.2: Largest singular values (left) and identified output (right) of [96-006]:Data of a laboratory setup acting like a hair dryer from DaiSy, with regularization parameter γ equals 2.5 (top), 1.75 (middle), 1.5 (bottom), respectively.

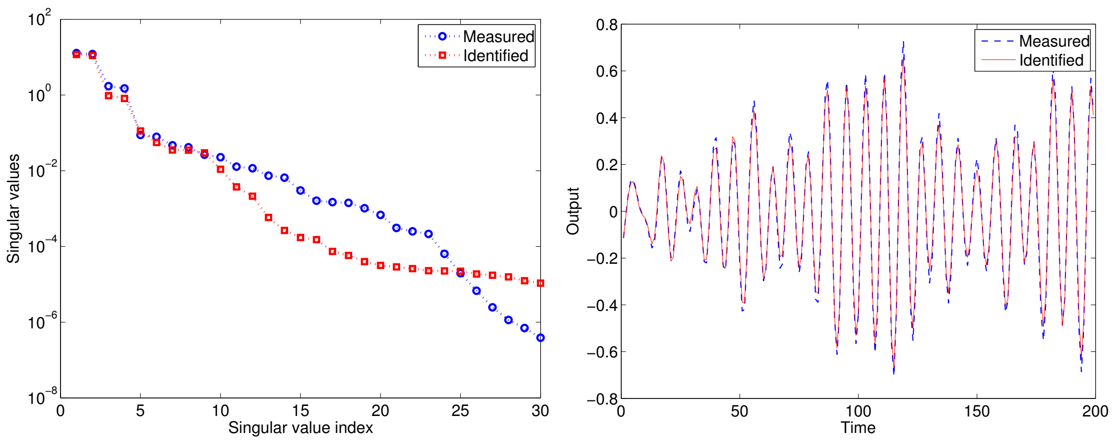


Figure 5.3: Largest singular values (left) and identified output (right) of [96-009]:Data from a flexible robot arm from DaiSy with γ equals 11.

CHAPTER 6

Conclusions

The thesis studies several first-order splitting methods that have been proposed for various applications, and applies them to a specific trace norm minimization problem.

Compared to an existing semidefinite programming solver, we see that as the problem data grows larger (with a couple hundreds primal variables), some first-order methods outperform the interior-point method in time and complexity with an acceptable accuracy, which is expected according to their computational complexity.

Among the studied first-order methods, we show that while ADMM has the reason for its long-lasting popularity, some other recent methods can be promising, too. For example, Pock-Chambolle's modified semi-implicit method shows fast convergence to a modest accuracy, and Tseng's modified forward-backward method is of interest when the optimal solution has relatively full rank.

We also see that trace norm minimization of a properly constructed matrix, with a proper choice of the quadratic regularization parameter, can often provide us a good estimate of the system order in system identification, which is especially helpful if the data is noise-polluted.

For possible future research directions, it is interesting to learn what makes some methods work well on the trace norm minimization problem, and to develop a general solver for it. As for the nature of using trace norm as an approximation of the rank, some interesting questions are: when is the approximation exact, that is, when is a low-rank solution the optimal one, what is the role of the regularization parameter and what determines the value that gives optimal fit in the setting of system identification and realization? Lastly, we would like to know if there are other convex heuristics for the rank minimization, for example, other norms like Ky Fan k-norms (k largest singular values) or Schatten m-norms.

References

[AH58] K. J. Arrow and L. Hurwicz. *Studies in Linear and Non-Linear Programming.* Stanford University Press, 1958.

[BPC11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. "Distributed optimization and statistical learning via the alternating direction method of multipliers." *Found. Trends Mach. Learn.*, **3**(1):1–122, 2011.

[BT09] A. Beck and M. Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems." *SIAM Journal on Imaging Sciences*, **2**(1):183–202, 2009.

[BV04] S. P. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[CP11] A. Chambolle and T. Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging." *J. Math. Imaging Vis.*, **40**(1):120–145, May 2011.

[CT94] G. Chen and M. Teboulle. "A proximal-based decomposition method for convex minimization problems." *Mathematical Programming*, **64**:81–101, 1994.

[DDD94] B. De Moor, P. De Gersem, B. De Schutter, and W. Favoreel. "DaISy: Database for the Identification of Systems.", 1994.

[DR56] J. Douglas and H. H. Rachford. "On the numerical solution of heat conduction problems in two and three space variables." *Transactions of the American mathematical Society*, **82**(2):421–439, 1956.

[DSC07] T. Ding, M. Sznaier, and O. Camps. "A rank minimization approach to fast dynamic event detection and track matching in video sequences." In *Decision and Control, 2007 46th IEEE Conference on*, pp. 4122–4127, 2007.

[EB92] J. Eckstein and D. P. Bertsekas. "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators." *Mathematical Programming*, **55**:293–318, 1992.

[FHB01] M. Fazel, H. Hindi, and S. P. Boyd. "A rank minimization heuristic with application to minimum order system approximation." *Proceedings of the American Control Conference*, 2001.

[FPS11] M. Fazel, T. K. Pong, D. Sun, and P. Tseng. "Hankel matrix rank minimization with applications in system identification and realization.", 2011.

[Gab83] D. Gabay. "Applications of the Method of Multipliers to Variational Inequalities." In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*, volume 15 of *Studies in Mathematics and Its Applications*, chapter IX, pp. 299 – 331. Elsevier, 1983.

[GJM09] C. Grossmann, C. N. Jones, and M. Morari. "System identification via nuclear norm regularization for simulated moving bed processes from incomplete data sets." In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. Proceedings of the 48th IEEE Conference on*, pp. 4692–4697, 2009.

[HJ85] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[HJ91] R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, 1991.

[LHV13] Z. Liu, A. Hansson, and L. Vandenberghe. "Nuclear norm system identification with missing inputs and outputs." *Submitted to System and Control Letters*, 2013.

[Lju99] L. Ljung. *System Identification: Theory for the User.* Prentice Hall, 2nd edition, 1999.

[LM79] P. Lions and B. Mercier. "Splitting algorithms for the sum of two nonlinear operators." *SIAM Journal on Numerical Analysis*, **16**(6):964–979, 1979.

[LV09] Z. Liu and L. Vandenberghe. "Interior-point method for nuclear norm approximation with application to system identification." *SIAM Journal on Matrix Analysis and Applications*, **31**(3):1235–1256, 2009.

[MGC11] S. Ma, D. Goldfarb, and L. Chen. "Fixed point and Bregman iterative methods for matrix rank minimization.", 2011.

[PCB09] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. "An algorithm for minimizing the Mumford-Shah functional." In *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 1133–1140, 2009.

[RFP10] B. Recht, M. Fazel, and P. A. Parrilo. "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization." *SIAM Review*, (3):471–501, 2010.

[Roc76] R. T. Rockafellar. "Monotone operators and the proximal point algorithm." *SIAM Journal on Control and Optimization*, **14**(5):877–898, 1976.

[Roc97] R. T. Rockafellar. *Convex Analysis.* Princeton Landmarks in Mathematics, 1997.

[Tse91] P. Tseng. "Applications of a splitting algorithm to decomposition in convex programming and variational inequalities." *SIAM Journal on Control and Optimization*, **29**(1):119–138, 1991.

[Tse00] P. Tseng. "A modified forward-backward splitting method for maximal monotone mappings." *SIAM Journal on Control and Optimization*, **38**(2):431–446, 2000.

[VB96] L. Vandenberghe and S. P. Boyd. "Semidefinite programming." *SIAM Review*, (1):49–95, 1996.

[VV07] M. Verhaegen and V. Verdult. *Filtering and System Identification: A Least Squares Approach.* Cambridge University Press, 2007.