Titre: Title:	A Regularized Interior-Point Method for Constrained Linear Least Squares
Auteur: Author:	Mohsen Dehghani
Date:	2013
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Dehghani, M. (2013). A Regularized Interior-Point Method for Constrained Linear Least Squares [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1121/

UNIVERSITÉ DE MONTRÉAL

A REGULARIZED INTERIOR-POINT METHOD FOR CONSTRAINED LINEAR LEAST SQUARES

MOHSEN DEHGHANI
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
AVRIL 2013

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

A REGULARIZED INTERIOR-POINT METHOD FOR
CONSTRAINED LINEAR LEAST SQUARES

présenté par : DEHGHANI Mohsen
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury constitué de :

M. LE DIGABEL Sébastien, Ph.D., président
M. ORBAN Dominique, Doct. Sc., membre et directeur de recherche
M. VIEIRA Manuel Valdemar Cabral, Doctorat, membre

To Who has taught by the pen. He has taught man that which he knew not...
(From the Holy Quran, Surah : 96, AL-Alaq, Verse : 4-5)

ACKNOWLEDGEMENT

This research project would not have been possible without the support of many people. The author wishes to express his gratitude to his supervisor, Prof. Dr. Dominique Orban who was abundantly helpful and offered invaluable assistance, support and guidance. Deepest gratitude is also due to the members of the Jury committee, Prof. Dr. Le Digabel Sébastien and Prof. Vieira Manuel, and his brother Ahad Dehghani without whose knowledge and assistance this study would not have been successful. Special thanks also to all my friends, especially my GERAD friends for sharing invaluable assistance.

The author wishes to express his love and gratitude to his beloved family members for their understanding throughout the duration of his studies.

RÉSUMÉ

Nous proposons une méthode de points intérieurs non réalisable pour le problème aux moindres carrés linéaire avec contraintes basée sur la régularisation primale-duale de problèmes quadratiques convexes de Friedlander and Orban (2012). À chaque itération, la méthode effectue une factorisation LDL^T creuse d'une matrice symétrique et quasi définie. Cette matrice est uniformément bornée et non singulière. Nous établissons des conditions sous lesquelles la méthode produit une solution du problème original. La régularisation nous permet d'éliminer l'hypothèse que les gradients actifs sont linéairement indépendants. Bien que l'implémentation proposée ici repose sur une factorisation, elle ouvre la voie à une implémentation itérative dans laquelle on résout un problème aux moindres carrés régularisé sans contraintes de façon inexacte à chaque itération. Nous illustrons notre approche sur plusieurs applications qui mettent en évidence ses avantages.

ABSTRACT

We propose an infeasible interior-point algorithm for constrained linear least-squares problems based on the primal-dual regularization of convex programs of Friedlander and Orban (2012). At each iteration, the sparse LDL^T factorization of a symmetric quasi-definite matrix is computed. This coefficient matrix is shown to be uniformly bounded and nonsingular. We establish conditions under which a solution of the original problem is recovered. The regularization allows us to dispense with the assumption that the active gradients are linearly independent. Although the implementation described here is factorization based, it paves the way for a matrix-free implementation in which a regularized unconstrained linear least-squares problem is solved at each iteration. We report on computational experience and illustrate the potential advantages of our approach.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

LS	Least-Squares
DEM	Direct Elimination Method
NLS	Non-Linear Least-Squares
SQD	Symmetric Quasi-Definite
QP	Quadratic Program(ing)
LP	Linear Program(ing)
SVD	Singular Value Decomposition
KKT	Karush-Kuhn-Tucker
CG	Conjugate Gradient Method
LSQR	Sparse Equations and Least Squares
CGNE	Conjugate Gradient Method On the Normal Equations
CGLS	Conjugate Gradient Method On the Least-Squares
NMF	Non-negative Matrix Factorization
DCT2	2-D Discrete Cosine Transform

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

We are concerned with the constrained linear least-squares problem in standard form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ c^Tx + \tfrac{1}{2}\|Ax-d\|^2 \quad \text{subject to}\ \ Bx = b,\ x \geq 0, \tag{1.1}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{p\times n}$, $d \in \mathbb{R}^p$, $B \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and inequalities are understood componentwise. It is typically assumed that $p > n$ and $m < n$ but the approach proposed in this document allows us to do away with these restrictions. If $A = 0$, (1.1) reduces to the linear programming problem in standard form. In all other cases, (1.1) is a convex quadratic program. An interior-point method applied directly to (1.1) might suffer several difficulties. Firstly, the matrix A^TA, which may be rather dense, will appear explicitly in the Newton step computation. Secondly, numerical instabilities will arise if the constraint matrix B does not have full row rank. We remove the first difficulty in two different ways that lead to two slightly different implementations. The second difficulty disappears by considering the following regularization of (1.1) proposed by Friedlander and Orban (2012):

$$\begin{aligned} \underset{x\in\mathbb{R}^n,w\in\mathbb{R}^m}{\text{minimize}} \quad & c^Tx + \tfrac{1}{2}\|Ax-d\|^2 + \tfrac{1}{2}\rho\|x-x_k\|^2 + \tfrac{1}{2}\delta\|w+y_k\|^2 \\ \text{subject to} \quad & Bx + \delta w = b,\ x \geq 0, \end{aligned} \tag{1.2}$$

where $\rho > 0$ and $\delta > 0$ are regularization parameters, x_k and y_k are the current approximations of the optimal primal variables and Lagrange multipliers, respectively, and w are auxiliary variables playing the role of a constraint residual. In this document, we specialize the interior-point framework of Friedlander and Orban (2012) and apply it to (1.2) with ultimately constant regularization parameters. At each iteration, a step is computed by solving a large and sparse symmetric quasi-definite linear system (Vanderbei, 1995). Contrary to most interior-point implementations, partial block elimination is not applied to this system to reduce it to the so-called *augmented system* form or to the *normal equations*. Instead, a similarity transformation is applied that guarantees that the system remains uniformly bounded and nonsingular throughout the iterations and in the limit provided strict complementarity is satisfied at a solution. We establish global convergence under weak assumptions. In particular, no assumption on the rank of B or A is made. A distinctive feature of the regularization (1.2) is that it allows to recover a solution of (1.1) in many situations and

not only a solution of a perturbed problem. In addition, (1.2) is never solved to optimality for fixed values of ρ, δ, x_k and y_k. Instead, it is used to compute a single Newton step before attention turns to the next regularized subproblem. In (1.2), the *primal* regularization term $\frac{1}{2}\rho\|x - x_k\|^2$ serves the dual purpose of regularizing A whenever it is rank deficient and simplifying the implementation of the interior-point method in the presence of free variables. The *dual* regularization term $\frac{1}{2}\delta\|w + y_k\|^2$ regularizes B whenever it is rank deficient. The implementation proposed below relies on a sparse LDL^T factorization of the symmetric quasi-definite coefficient matrix. This factorization may be obtained at lower cost than the symmetric indefinite factorization, such as that of Duff (2004), and typically yields sparser factors. Its stability on symmetric quasi-definite systems has been analyzed by Gill et al. (1996). Many applications only provide A and B in the form of linear operators instead of explicit matrices. Iterative methods specialized to symmetric quasi-definite systems have been recently proposed by Arioli and Orban (2012). Our algorithm paves to way to a matrix-free implementation using such iterative methods. This yields an elegant framework in which an unconstrained regularized linear least-squares problem must be solved at each iteration. Our analysis and implementation differ from those of Friedlander and Orban (2012) in several respects. Firstly, the linear systems used in the definition of the Newton steps are larger, sparser and tailored to the special structure of (1.1). If strict complementarity holds at a solution, they also have uniformly bounded condition number. Secondly, our approach illustrates how to apply the primal-dual regularization of Friedlander and Orban (2012) selectively, leaving some variables and some constraints untouched. This has the benefit of exploiting the structure of the problem at hand.

1.0.1 Notation

The notation X and Z is used to denote the diagonal matrices $\mathrm{diag}(x)$ and $\mathrm{diag}(z)$. The vector e denotes the vector of all ones of appropriate dimension. The notation $\|\cdot\|$ denotes the Euclidian norm throughout. The i-th component of a vector x is denoted $[x]_i$ while the value of x at the k-th iteration of a process is denoted x_k. For a given positive definite matrix M, the M-norm is defined as $\|x\|_M^2 = x^T M x$. The notation $\mathrm{blkdiag}(A_1, \ldots, A_k)$ denotes a block-diagonal matrix having the blocks A_1 through A_k consecutively on the diagonal. Whenever a block A_j is an identity block, its size is dictated by the context. For two related sequences $\{\alpha_k\}$ and $\{\beta_k\}$ of positive numbers, we write $\alpha_k = O(\beta_k)$ if there exists a constant $C > 0$ such that $\alpha_k \leq C\beta_k$ for all sufficiently large k. We write $\alpha_k = \Theta(\beta_k)$ if $\alpha_k = O(\beta_k)$ and $\beta_k = O(\alpha_k)$.

CHAPTER 2

LITERATURE REVIEW

Carl Friedrich Gauss is credited with developing the fundamentals of the basis for least-squares (LS) analysis in 1795 at the age of eighteen. Gauss's method came to be on January 1, 1801. The modern approach was first exposed in 1805 by the French mathematician Legendre Levenberg (1944). Nowadays, the LS method is widely used to find or estimate the numerical values of parameters to fit a function to a set of data. In this thesis, we are concerned with constrained and unconstrained linear least-squares problems. The second is the (Un)constrained Non-Linear Least-Squares method (NLS). We shall explain the NLS problem in a future section. There are essentially three different families of algorithms for solving a unconstrained linear-least square problem:

1. methods based on the normal equations;
2. methods based on the QR factorization;
3. methods based on the singular-value decomposition (SVD).

The first approach is the fastest and the most sensitive to ill conditioning. On the other hand, SVD is the most expensive and most accurate. Using the QR factorization to solve LS is numerically stable. An overview of those families of methods is provided in the following sections.

2.1 Factorization

The matrix factorization is a very useful linear algebra transformation, which targets the presentation of a matrix A as an appropriate product of matrices. There are many different matrix decompositions. Each finds use among a particular class of problems. Here we introduce a summary of the important matrix factorizations.

2.1.1 QR Factorization

The QR factorization decomposes the matrix $A \in \mathbb{R}^{m \times n}$ as

$$A = QR, \tag{2.1}$$

where R is upper triangular and Q is orthogonal, i.e., $Q^T Q = I$. Such a decomposition can be performed both for a square matrix A with dimensions $n \times n$, as well as more general

rectangular A with dimensions $m \times n$. To solve the linear system of equations $Ax = b$, firstly the vector $Q^T b = b_Q$ is evaluated and then the triangular linear system $Rx = b_Q$ is solved. Due to the upper triangular form of R, this system of linear equations is easily solved by *back substitution*. The standard algorithm for the QR decomposition involves a sequence of *Householder transformations*. In the following sections, we provide more details about the answers to the following questions:

1. Does the QR factorization always exist?
2. Is this factorization unique?
3. Which Algorithms perform the QR factorization?
4. How is the QR factorization useful for solving linear least-squares problems?
5. What codes are available to perform the QR factorization?

2.1.1.1 Existence of the QR Factorization

For every matrix A, a QR factorization exists, even if A does not have full rank. The existence of this factorization follows from *Householder transformations*. One can prove the following two theorems related to the QR factorization of a matrix A.

Theorem 2.1.1. *Every matrix A possesses a QR factorization.*

Theorem 2.1.2. (Full QR Factorization) *(Trefethen and Bau (1997)[Theorem 5.1]) Let A be a non-singular matrix. There exists a* **unique** *pair (Q, R), where Q is an orthogonal matrix and R is an upper triangular matrix, whose diagonal entries are real, satisfying $A = QR$.*

The overall complexity (number of oating points) of the QR factorization is n^3

2.1.1.2 Forms of the QR Factorization

– **If A has full rank**

1. If A is square, R has the form

$$R_{n,n} = \begin{pmatrix} \star & \star & \cdots & \star \\ 0 & \star & \cdots & \star \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \star \end{pmatrix}.$$

If A is non-square, R is non-square too,

2. If A has more columns than rows, i.e., $n > m$ we can write $R = \begin{bmatrix} R_1 & | & R_2 \end{bmatrix}$ where R_1 is upper triangular.
3. The most common case encountered in linear least-squares problems is where A is $m \times n$, with $m > n$. In this case we have $R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}$ where R_1 is $n \times n$ triangular, and

$$\begin{aligned} Q &= \begin{bmatrix} Q_1 & | & Q_2 \end{bmatrix} \\ A &= \begin{bmatrix} Q_1 & | & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \\ &= Q_1 R_1 + Q_2 0 \\ &= Q_1 R_1, \end{aligned}$$

where Q_1 is an $m \times n$ matrix whose columns are orthogonal.

– **If A does have not full rank**
In this case R has the form

$$R = \begin{pmatrix} \star & \star & \cdots & \star \\ & \star & \cdots & \star \\ & & \ddots & \vdots \\ & \mathbf{0} & & \star \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

It is often of interest to discover the rank of A. Given a decomposition of the form (2.1), $\text{rank}(A) = \text{rank}(R)$, and in practice, this QR decomposition is a good way to determine the rank of a matrix. The computations are quite sensitive to rounding, however, and therefore it must be done with some care. If columns of A are linearly independent, then this factorization is unique. There are many practical algorithms in Golub and Van Loan (1996).

2.1.1.3 Codes Available to Perform the QR Factorization

1. BAND–QR is a FORTRAN90 library which includes LAPACK–style routines to compute the QR factorization of a banded matrix.
2. From python using the **scipy** package, we can use scipy.linalg.qr.
3. Using **qr** from Matlab.

2.1.2 SVD

The Singular Value Decomposition (SVD) of a matrix A takes the form,

$$A = U\Sigma V,$$

where U and V are orthogonal and Σ is a diagonal positive semidefinite matrix.

Theorem 2.1.3. (The Singular Value Decomposition Theorem) *Let A be a real $m \times n$ matrix. Then there exist orthogonal matrices U and V such that*

$$U^T A V = \begin{bmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{bmatrix} = \Sigma,$$

where Σ_1 is a nonsingular diagonal matrix. The diagonal entries of Σ are all nonnegative and can be arranged in a non increasing order. The number of nonzero diagonal entries of Σ equals the rank of A.(Datta (2010)[Theorem 10.2.1*])*

The SVD of a matrix A is typically computed by a two-step procedure. In the first step, the matrix is reduced to a bidiagonal matrix. This takes $\mathcal{O}(mn^2)$ floating-point operations (flops), assuming that $m \geq n$ (this formulation uses the big $\mathcal{O}$ notation). The second step is to compute the SVD of the bidiagonal matrix. This step can only be done with an iterative method (as with eigenvalue algorithms Trefethen and Bau III 1997, Lecture 31).

2.1.3 Cholesky Factorization

Theorem 2.1.4. (The Cholesky Factorization Theorem) *If A is symmetric and positive definite, then A can be decomposed as*

$$A = LL^T$$

where L is a lower triangular matrix with strictly positive diagonal entries. (Trefethen and Bau (1997)[Theorem 32.1.*])*

The cost of computing the Cholesky factorization is $1/3n^3$ flops if A is of order n.

2.1.4 SQD Matrix and LDL^T Factorization

A symmetric matrix is called quasidefinite if it can be written, perhaps after a symmetric permutation, as

$$\begin{bmatrix} -E & A \\ A^T & D \end{bmatrix},$$

where E and D are symmetric and positive definite matrices. The LDL^T factorization can be used on this kind of matrix. The following theorem states a nice feature of SQD matrices. This property can be used to have a sparse L in the LDL^T factorization of a SQD matrix.

Theorem 2.1.5. *Any symmetric permutation P of a SQD matrix K possesses a factorization $PKP^T = LDL^T$, where L is unit lower triangular and D is diagonal. (Vanderbei (1995)[Theorem 12])*

2.2 Unconstrained Linear Least-Squares Problems

Suppose $A \in \mathbb{R}^{p\times n}, p \geq n, d \in \mathbb{R}^p$ are given. We can write the residual vector as

$$r(x) = Ax - d \quad \text{for } x \in \mathbb{R}^n.$$

The linear least-squares problem is defined as the following optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{2.2}$$

where $f(x) = \frac{1}{2}\|r(x)\|^2 = \frac{1}{2}\|Ax - d\|^2$. By definition of $f(x)$ the gradient and the Hessian of $f(x)$ are $\nabla f(x) = A^T(Ax - d)$ and $\nabla^2 f(x) = A^T A$. Since $x^T A^T Ax = \|Ax\|^2 \geq 0$ for any $x \in \mathbb{R}^n$, $\nabla^2 f(x)$ is positive semi-definite. Therefore, $f(x)$ is convex and any point x^* for which $\nabla f(x^*) = 0$ is a global minimizer of f. Therefore a solution, x^* must satisfy the following linear system of equations:

$$A^T Ax^* = A^T d. \tag{2.3}$$

In other words, since $\nabla^2 f(x)$ is positive semi-definite, $\nabla f(x) = 0$ are not only necessary but also sufficient conditions for optimality. We call (2.3) the *normal equations* of (2.2).

2.2.0.1 Using the QR Factorization to Solve LS

The Householder implementation of the QR factorization requires $2mn^2 - \frac{2}{3}n^3$ flops. The Euclidean norm of any vector is not affected by orthogonal transformations. Therefore, we have

$$\|Ax - d\| = \|Q^T(Ax - d)\|, \tag{2.4}$$

for any $m \times m$ orthogonal matrix Q. Suppose we perform a QR factorization on matrix A, so that

$$A = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1, \tag{2.5}$$

where Q_1 is $n \times n$, Q_2 is $(m-n) \times n$ and R_1 is $n \times n$. From (2.4) and (2.5) we have

$$\begin{aligned}
&\|Ax - d\|^2 \\
&= \left\| \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} (Ax - d) \right\|^2 \\
&= \left\| \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix} \left(\begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} x - d \right) \right\|^2 \\
&= \left\| \begin{bmatrix} R_1 \\ 0 \end{bmatrix} x - \begin{bmatrix} Q_1^T d \\ Q_2^T d \end{bmatrix} \right\|^2 \\
&= \left\| \begin{bmatrix} R_1 x - Q_1^T d \\ 0 - Q_2^T d \end{bmatrix} \right\| \\
&= \|R_1 x - Q_1^T d\|^2 + \|Q_2^T d\|^2
\end{aligned}$$

The second term of the last expression is not dependent on x. If we want to minimize (2.2), the optimal solution is equal to

$$x^* = R_1^{-1} Q_1^T d,$$

and the optimal objective value is $\|Q_2^T d\|$. In summary, we can use the following procedure

1. Compute the reduced QR factorization $A = QR$.
2. Compute the vector $Q^T d$.
3. Solve the upper-triangular system $Rx = Q^T d$ for x.

Fore more information, see (Nocedal and Wright (1999)[10.2]).

2.2.0.2 Using the Cholesky Factorization to Solve LS

The classical way to solve least-squares problems is to solve the normal Equation (2.3). The standard method of solving (2.3) is to use the Cholesky factorization, $A^TA = LL^T$ where L is lower-triangular, reducing (2.3) to

$$LL^Tx = A^Td. \tag{2.6}$$

Now consider the factorization $A = QR$, then $A^TA = R^TR$. The uniqueness of the Cholesky factors then implies that $R = L^T$. In summary, we can use the following procedure:

1. Compute the coefficient matrix A^TA and the right-hand-side A^Td.
2. Compute the Cholesky factorization of the symmetric matrix $A^TA = LL^T = R^TR$.
3. Solve the lower-triangular system $Ly = A^Td$ for y.
4. Solve the upper-triangular system $L^Tx = y$ for x.

Computing A^TA requires mn^2 flops and the Cholesky factorization requires $\frac{n^3}{6}$ flops. All together, solving least-squares problems by the normal equations involve $mn^2 + \frac{1}{6}n^3$ flops. If $m \gg n$ this method is twice as fast as the QR factorization.

2.2.0.3 Using the SVD Factorization to Solve LS

Let $A = U\Sigma V^T$ be the SVD of A. Then we have

$$\begin{aligned} \|Ax - d\|_2 &= \|U\Sigma V^Tx - d\|_2 \\ &= \|U(\Sigma V^Tx - U^Td)\|_2 \\ &= \|\Sigma y - d'\|_2 \end{aligned}$$

where $V^Tx = y$ and $U^Td = d'$. Thus, the use of SVD of A reduces the least-squares problem for a full matrix A to one with a diagonal matrix Σ. Now we need to solve the following trivial optimization problem (Datta (2010)[10.2]).

$$\underset{y}{\text{minimize}} \quad \|\Sigma y - d'\|.$$

2.2.1 Iterative Methods to Solve LS

Linear least-squares problems can also be solved using iterative methods that generally fall into the category of Krylov methods Bjorck (1996). We give a brief overview of such methods in the rest of this section.

2.2.1.1 Conjugate Gradient Method on the Normal Equations (CGNE)

The conjugate gradient method can be applied to an arbitrary system $Ax = d$ by applying it to the normal equations matrix A^TA and right-hand side vector A^Td, since A^TA is a symmetric positive-semidefinite matrix for any A. The result is the conjugate gradient method on the normal equations (CGNE). As an iterative method, it is not necessary to form A^TA explicitly in memory; rather, only to perform matrix-vector and transpose matrix-vector multiplications. Therefore (CGNE) is particularly useful when A is a large sparse matrix since these operations are usually extremely efficient. However, the downside of forming the normal equations is that the condition number $Cond(A^TA)$ is equal to $Cond^2(A)$ and so the rate of convergence of (CGNE) may be slow and the quality of the approximate solution may be sensitive to roundoff errors. Finding a good preconditioner is often an important part of using the (CGNE) method. Several algorithms have been proposed (e.g., CGLS, LSQR). The LSQR algorithm purportedly has the best numerical stability when A is ill-conditioned, i.e., A has a large condition number (Lawson and Hanson (1995)[20]).

2.2.1.2 LSQR

LSQR is an algorithm for solving sparse least-squares problems. Consider the following regularized problem

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\left\| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} d \\ 0 \end{bmatrix} \right\|^2 \tag{2.7}$$

where A and d are given data and λ is an arbitrary real scalar. The matrix A may be square or rectangular over-determined or under-determined, and may have any rank. The solution of (2.7) satisfies the symmetric quasi-definite system

$$\begin{bmatrix} I & A \\ A^T & -\lambda^2 I \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix},$$

where r is the residual vector $d - Ax$. Paige and Saunders (1982) use an iterative method based on the bidiagonalization procedure of Golub and Kahan. LSQR is algebraically equivalent to applying CG to the normal equations $(A^TA)x = A^Td$, but has better numerical properties, especially if A is ill-conditioned (Paige and Saunders (1982)).

2.3 Constrained Linear Least-Squares Problems

In terms of linear least-squares with linear equality constraints, we have the following base methods. One could consider the LS problem as

$$\underset{x}{\text{minimize}} \quad c^T x + \tfrac{1}{2}\|Ax - d\|^2 \qquad \text{subject to} \quad Bx = b, \quad x \geq 0. \tag{2.8}$$

Background and suggestions for further reading can be found in the seminal book of Hanson and Lawson (1969), who have described three methods for solving (2.8) as follows:

1. methods based on the null space;
2. methods based on direct elimination;
3. methods based on weighted LS.

2.4 Regularization

In this thesis, we consider a primal-dual regularization of convex QPs which specializes the interior-point framework of Friedlander and Orban (2012). Consider the convex quadratic program (QP)

$$\underset{x}{\text{minimize}} \quad c^T x + \tfrac{1}{2}x^T Q x \qquad \text{subject to} \quad Ax = b, \quad x \geq 0. \tag{2.9}$$

Now consider the regularization

$$\underset{x,r}{\text{minimize}} \quad c^T x + \tfrac{1}{2}x^T Q x + \tfrac{1}{2}\rho\|x - x_k\|^2 + \tfrac{1}{2}\delta\|r + y_k\|^2 \qquad \text{subject to} \quad Ax + \delta r = b, \quad x \geq 0, \tag{2.10}$$

of (2.9), where $\rho > 0$ and $\delta > 0$ are regularization parameters, and x_k and y_k are current estimates of primal and dual solutions. The strength of the approach is that the dual of (2.10) is the regularization of the dual of (2.9). It is easy to see that a constrained linear least-squares problem is a special case of convex QP. Indeed, consider the following constrained linear least-squares problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \; c^T x + \tfrac{1}{2}\|Ax - d\|^2 \quad \text{subject to } Bx = b, \; x \geq 0. \tag{2.11}$$

It is equivalent to the following convex QP:

$$\begin{aligned} \underset{x,r}{\text{minimize}} \quad & c^T x + \tfrac{1}{2}\|r\|^2 \\ \text{subject to} \quad & Bx = b, \quad Ax + r = d, \quad x \geq 0. \end{aligned} \tag{2.12}$$

CHAPTER 3

A VARIANT OF THE METHOD OF FRIEDLANDER AND ORBAN (2012)

In this chapter we are going to specialize an infeasible regularized interior-point algorithm called primal-dual for constrained linear least-squares problems (2.8) based on the primal-dual regularization of convex programs (2.9) of Friedlander and Orban (2012). Our approach illustrates how to apply the primal-dual regularization of Friedlander and Orban (2012) selectively, leaving some variables and some constraints untouched.

3.0.1 Regularization in the Primal-Dual Interior-Point Method

Consider the regularized form of (2.11)

$$\begin{aligned} \underset{x,w}{\text{minimize}} \quad & c^T x + \tfrac{1}{2}\|Ax - d\|^2 + \tfrac{1}{2}\rho\|x - x_k\|^2 + \tfrac{1}{2}\delta\|w + y_k\|^2 \\ \text{subject to} \quad & Bx + \delta w = b, \quad x \geq 0, \end{aligned} \tag{3.1}$$

proposed by Friedlander and Orban (2012) where $\rho > 0$ and $\delta > 0$ are regularization parameters, and x_k and y_k are current estimates of primal and dual solutions. The original problem (2.11) can be recovered by considering $\rho = 0$ and $\delta = 0$. The dual is given by

$$\begin{aligned} \underset{x,y,s,z}{\text{maximize}} \quad & b^T y - (A^T d)^T x - \tfrac{1}{2}\|Ax - d\|^2 - \tfrac{1}{2}\delta\|y - y_k\|^2 - \tfrac{1}{2}\rho\|s + x_k\|^2 \\ \text{subject to} \quad & B^T y + z - A^T A x = c - A^T d, \; z \geq 0, \end{aligned} \tag{3.2}$$

where $\{y, z\}$ are Lagrange multipliers corresponding to the equalities and bound constraints of (2.11) and $s = x - x_k$ are auxiliary variables.

An interior-point method places the slacks in a barrier term, leading to the following primal-dual pair in which $\mu > 0$ is a barrier parameter:

$$\begin{aligned} \underset{x,w}{\text{minimize}} \quad & c^T x + \tfrac{1}{2}\|Ax - d\|^2 + \tfrac{1}{2}\rho\|x - x_k\|^2 + \tfrac{1}{2}\delta\|w + y_k\|^2 - \mu_k \sum_{i=1}^{n} \ln x_i \\ \text{subject to} \quad & Bx + \delta w = b, \end{aligned} \tag{3.3}$$

$$\begin{aligned} \underset{x,y,s,z}{\text{maximize}} \quad & b^T y - (A^T d)^T x - \tfrac{1}{2}\|Ax - d\|^2 - \tfrac{1}{2}\delta\|y - y_k\|^2 - \tfrac{1}{2}\rho\|s + x_k\|^2 - \mu_k \sum_{i=1}^{n} \ln z_i \\ \text{subject to} \quad & B^T y + z - A^T A x = c - A^T d. \end{aligned} \tag{3.4}$$

3.0.2 Newton System

Let (x_k, y_k) be temporarily fixed. A primal-dual interior point method applied to the regularized problems (3.3) and (3.4) is based on applying a single Newton iteration to a sequence of non-linear equations of the form

$$\omega_k(v; \rho, \delta, \mu_k) := \begin{bmatrix} c + \rho s - A^T r - B^T y - z \\ Ax + r - d \\ \rho x - \rho(s + x_k) \\ \delta y - \delta(w + y_k) \\ Bx + \delta w - b \\ Xz - \sigma\mu_k e \end{bmatrix} = 0, \quad (x, z) \geq 0, \tag{3.5}$$

where $v = (x, r, s, w, y, z)$, $\mu_k > 0$ is the current duality measure, which is equal to $x_k^T z_k / n$, and $\sigma \in [0, 1]$ is a centring parameter. For fixed ρ_k, δ_k, x_k, and y_k the central path is the exact solution of (3.5) with $\sigma = 1$ (Wright, 1997). As $\mu_k \to 0$, it can be illustrated that this central path leads to a primal-dual solution to (3.3) and (3.4). Since objective function and constraints are convex, the necessary and sufficient optimality conditions can be written more succinctly as

$$\omega(v; 0, 0, 0) = 0, \text{ and } (x, z) \geq 0.$$

A Newton step for (3.5) from the current iterate ω_k is based on solving the system

$$\begin{bmatrix} 0 & -A^T & \rho I & 0 & -B^T & -I \\ A & I & 0 & 0 & 0 & 0 \\ \rho I & 0 & -\rho I & 0 & 0 & 0 \\ 0 & 0 & 0 & -\delta I & \delta I & 0 \\ B & 0 & 0 & \delta I & 0 & 0 \\ Z & 0 & 0 & 0 & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta s \\ \Delta w \\ \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} c + \rho s - A^T r - B^T y - z \\ Ax + r - d \\ \rho x - \rho(s + x_k) \\ \delta y - \delta(w + y_k) \\ Bx + \delta w - b \\ Xz - \sigma\mu e \end{bmatrix}. \tag{3.6}$$

The block matrix in (3.6) can be reduced by eliminating the variables Δw and Δs:

$$\begin{bmatrix} -\rho I & A^T & B^T & I \\ A & I & 0 & 0 \\ B & 0 & \delta I & 0 \\ Z & 0 & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} c - A^T r - B^T y - z \\ d - Ax - r \\ b - Bx \\ XZ - \sigma\mu e \end{bmatrix}. \tag{3.7}$$

The remaining directions may be recovered via

$$\Delta w = \Delta y - w_k, \qquad \Delta s = \Delta x - s_k. \tag{3.8}$$

By eliminating the variable Δz from (3.7) we arrive at

$$\begin{bmatrix} -(X^{-1}Z + \rho I) & A^T & B^T \\ A & I & 0 \\ B & 0 & \delta I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta y \end{bmatrix} = \begin{bmatrix} c - A^T r - B^T y - \sigma\mu X^{-1} e \\ d - Ax - r \\ b - Bx \end{bmatrix}, \tag{3.9}$$

where the variable Δz is recovered via

$$\Delta z = -z - X^{-1} Z \Delta x + \sigma\mu X^{-1} e. \tag{3.10}$$

The system (3.9) will be discussed further in this chapter. Note that upon setting $\rho = \delta = 0$, we recover the Newton equations used to compute a step from the k-th iterate of an interior-point method applied to (3.1)–(3.2). As shown in (3.9) we take $x = x_k$, $r = r_k$, and $y = y_k$ at each iteration. More precisely, the central path $\mathcal{C}$ is an arc of strictly feasible points defined as the solutions of

$$Bx + \delta w - b = 0 \tag{3.11a}$$
$$Ax + r - d = 0 \tag{3.11b}$$
$$c - A^T r + \rho s - B^T y - z = 0 \tag{3.11c}$$
$$\delta y - \delta(w + y_k) = 0 \tag{3.11d}$$
$$\rho x - \rho(s + x_k) = 0 \tag{3.11e}$$
$$Xz = \mu e \tag{3.11f}$$
$$(x, z) > 0, \tag{3.11g}$$

for positive values of μ. If $(x_\mu, r_\mu, s_\mu, w_\mu, y_\mu, z_\mu)$ solves (3.11) then the central path is the set

$$\mathcal{C} = \{(x_\mu, r_\mu, s_\mu, w_\mu, y_\mu, z_\mu) \quad | \quad \mu > 0\}.$$

It can be shown that $(x_\mu, r_\mu, s_\mu, w_\mu, y_\mu, z_\mu)$ is defined uniquely for any $\mu > 0$ if and only if $\mathcal{C}$ is non-empty (Wright, 1997, Theorem 2.8).

3.0.3 $\mathcal{N}_k$ Neighbourhood

A difference between our approach and traditional interior-point methods is that during the course of the iterations, the regularization parameters ρ and δ may be updated. At the k-th iteration, the current iterate is $v_k := (x_k, r_k, s_k, w_k, y_k, z_k)$ and the regularization parameters have values ρ_k and δ_k. We consider a neighbourhood $\mathcal{N}_k$ of the central path as the set of points (x, r, s, w, y, z) that satisfy the following conditions:

$$\bar{\gamma}_C x^T z/n \geq [x]_i[z]_i \geq \gamma_C x^T z/n, \tag{3.12a}$$

$$x^T z \geq \gamma_P \|Bx + \delta_k w - b\|, \tag{3.12b}$$

$$x^T z \geq \gamma_R \|Ax + r - d\|, \tag{3.12c}$$

$$x^T z \geq \gamma_D \|c + \rho_k s - B^T y - A^T r - z\|, \tag{3.12d}$$

$$x^T z \geq \gamma_S \|\rho_k x - \rho_k(s + x_k)\|, \tag{3.12e}$$

$$x^T z \geq \gamma_W \|\delta_k y - \delta_k(w + y_k)\|, \tag{3.12f}$$

where $0 < \gamma_C < 1 < \overline{\gamma}_C$ and $(\gamma_P, \gamma_D, \gamma_R, \gamma_S, \gamma_W) > 0$ are given constants. Our interior-point scheme generates the new iterate v_{k+1} as follows:

$$v_k(\alpha_k) := (x_k + \alpha_k \Delta x, r_k + \alpha_k \Delta r, s_k + \alpha_k \Delta s, w_k + \alpha_k \Delta w, y_k + \alpha_k \Delta y, z_k + \alpha_k \Delta z),$$

where $\alpha_k \in (0, 1]$ and $(\Delta x, \Delta r, \Delta s, \Delta w, \Delta y, \Delta z)$ is computed via (3.9), (3.8), and (3.10).

3.0.4 Algorithm

Our algorithm is the same as (Friedlander and Orban, 2012, Algorithm 4.1) with the exception of the linear system used in Step 2, and is formalized as Algorithm 3.0.1.

Algorithm 3.0.1 Primal-Dual Regularized Interior-Point Algorithm

Step 0 [**Initialize**] Choose minimum and maximum centering parameters $0 < \sigma_{\min} \leq \sigma_{\max} < 1$, a constant $\sigma_{max} < \beta < 1$, proximity parameters $0 < \gamma_C < 1 < \bar{\gamma}_C$ and $(\gamma_P, \gamma_R, \gamma_D, \gamma_S, \gamma_W) > 0$, initial regularization parameters $\rho_0 > 0$, $\delta_0 > 0$, and a stopping tolerance $\epsilon > 0$. Let the neighborhood of the central path be defined by (3.12a), (3.12b), (3.12c) and (3.12d). Choose initial primal $x_0 \in \mathbb{R}^n_{++}$, $r_0 \in \mathbb{R}^m$, $w_0 \in \mathbb{R}^m$ and dual guesses $s_0 \in \mathbb{R}^n$, $y_0 \in \mathbb{R}^m$, and $z_0 \in \mathbb{R}^n_{++}$ so that $v_0 \in \mathcal{N}_0$. Set $\mu_0 := x_0^T z_0/n$ and $k = 0$.

Step 1 [**Test convergence**] If $x_k^T z_k \leq \epsilon$, declare convergence.

Step 2 [**Step computation**] Choose a centering parameter $\sigma_k \in [\sigma_{\min}, \sigma_{\max}]$. Compute the Newton step Δv_k from v_k, e.g., by solving (3.9) with and recovering the remaining components from (3.8) and (3.10).

Step 3 [**Linesearch**] Select $\delta_{k+1} \in (0, \delta_k]$ and $\rho_{k+1} \in (0, \rho_k]$ and compute α_k as the largest $\alpha \in (0, 1]$ such that

$$v_k(\alpha) \in \mathcal{N}_{k+1} \quad \text{and} \quad \mu_k(\alpha) \leq (1 - \alpha(1 - \beta))\mu_k, \tag{3.13}$$

where $\mu_k(\alpha) := x_k(\alpha)^T z_k(\alpha)/n$.

Step 4 [**Update iterates**] Set $v_{k+1} := v_k(\alpha_k)$, $\mu_{k+1} := \mu_k(\alpha_k)$. Increment k by 1 and go to Step 1.

3.1 Global Convergence Analysis

We begin our analysis by providing bounds on the eigenvalues of the matrix

$$K_k := \begin{bmatrix} -H_k & \tilde{A}^T \\ \tilde{A} & I_{\delta_k} \end{bmatrix}, \tag{3.14}$$

where

$$H_k = (X_k^{-1} Z_k + \rho_k I), \quad \tilde{A} = \begin{bmatrix} A \\ B \end{bmatrix}, \text{ and } I_{\delta_k} = \begin{bmatrix} I & 0 \\ 0 & \delta_k I \end{bmatrix}.$$

The block matrix K_k in (3.14) is the matrix that appears in Step 2 of Algorithm 3.0.1. In the remainder of this section, we simplify the notation by dropping the iteration counter k.

The inertia of a symmetric matrix K is the triple of non-negative integers (n_+, n_-, n_0), where n_-, n_0, and n_+ are the number of negative, zero, and positive eigenvalues of K,

respectively. The following lemma states the relation between the inertia of matrix K and that of B^TKB where $B \in \mathbb{R}^{n\times n}$ is nonsingular.

Lemma 3.1.1. *(Sylvester's Law of Inertia) If $K \in \mathbb{R}^{n\times n}$ is symmetric and $B \in \mathbb{R}^{n\times n}$ is nonsingular, then K and B^TKB have the same inertia.*

Proof. See (Golub and Van Loan, 1996, Theorem 8.1.17). □

The eigenvalue bounds

$$\lambda_{\min}(H) \geq \lambda_{\min}(\tilde{A}) + \min_{1\leq i\leq n} \frac{[z]_i}{[x]_i} + \rho \quad \text{and} \quad \lambda_{\max}(H) \leq \lambda_{\max}(\tilde{A}) + \max_{1\leq i\leq n} \frac{[z]_i}{[x]_i} + \rho, \tag{3.15}$$

and the congruence relation

$$\begin{bmatrix} -H & \tilde{A}^T \\ \tilde{A} & I_\delta \end{bmatrix} = \begin{bmatrix} I & 0 \\ -\tilde{A}H^{-1} & I \end{bmatrix} \begin{bmatrix} -H & 0 \\ 0 & \tilde{A}H^{-1}\tilde{A}^T + I_\delta \end{bmatrix} \begin{bmatrix} I & -H^{-1}\tilde{A}^T \\ 0 & I \end{bmatrix} \tag{3.16}$$

are useful for the next theorem, which is a variant of (Friedlander and Orban, 2012, Theorem 5.1) for least-squares problems.

Theorem 3.1.2. *For all $(x, z) > 0$ and all $\rho > 0$ and $\delta > 0$, K possesses precisely n negative eigenvalues and $m + n$ positive eigenvalues. Let them be denoted and ordered as*

$$-\lambda_{-n} \leq \lambda_{-n+1} \leq \cdots \leq \lambda_{-1} < 0 < \lambda_1 \leq \cdots \leq \lambda_{m+n-1} \leq \lambda_{n+m}.$$

The largest positive and smallest negative eigenvalues of K satisfy the bounds

$$\lambda_{-n} \geq \frac{1}{2}[\delta - \lambda_{\max}(H)] - \frac{1}{2}[(\lambda_{\max}(H) - \delta)^2 + 4(\sigma_{\max}(\tilde{A})^2 + \lambda_{\max}(H)\delta)]^{\frac{1}{2}}, \quad (3.17a)$$

$$\lambda_{n+m} \leq \frac{1}{2}[\delta - \lambda_{\min}(H)] + \frac{1}{2}[(\lambda_{\min}(H) - \delta)^2 + 4(\sigma_{\min}(\tilde{A})^2 + \lambda_{\min}(H)\delta)]^{\frac{1}{2}}. \quad (3.17b)$$

The smallest positive and largest negative eigenvalues of K satisfy the bounds

$$\lambda_{-1} \leq -\lambda_{\min}(H), \quad (3.17c)$$

$$\lambda_1 \geq \frac{1}{2}[\delta + \lambda_{\max}(H)] - \frac{1}{2}[(\lambda_{\max}(H) - \delta)^2 + 4(\sigma_{\min}(\tilde{A})^2 + \lambda_{\max}(H)\delta)]^{\frac{1}{2}}. \quad (3.17d)$$

Moreover, $\lambda_1 = \tilde{\delta}$ where $\tilde{\delta} = \min(1, \delta)$ is the smallest positive eigenvalue of K if and only if $\tilde{A}$ does not have full row rank. In this case, its associated eigenspace is $\{0\} \times Null(\tilde{A}^T)$. Its geometric multiplicity is thus m+n-rank($\tilde{A}$).

Proof. The first part of the theorem follows from (3.16) and Sylvester's law of inertia. Note that H and $\tilde{A}H^{-1}\tilde{A}^T + I_\delta$ are positive definite because of the positivity assumption on x, z, ρ, and δ. The rest of the proof parallels (Silvester and Wathen, 1994, Lemma 2.2).

If $(u, v) \neq 0$ is an eigenvector of K associated to the eigenvalue λ, then

$$-Hu + \tilde{A}^T v = \lambda u \quad (3.18a)$$

$$\tilde{A}u + I_\delta v = \lambda v. \quad (3.18b)$$

Note that $\lambda = 1$ is an eigenvalue of K if and only if A does not have full row rank, and its associated eigenspace is $\{0\} \times Null(A^T)$, similarly, $\lambda = \delta$ is an eigenvalue of K if and only if B does not have full row rank, and its associated eigenspace is $\{0\} \times \{0\} \times Null(B^T)$ Now, suppose $\lambda \neq \delta$ and $\lambda \neq 1$. From (3.18b), we have $v = (\lambda I - I_\delta)^{-1}\tilde{A}u$. Necessarily, $u \neq 0$. Substituting into (3.18a) and taking the inner product with u yields

$$\lambda\|u\|^2 = -u^T Hu + (\lambda I - I_\delta)^{-1}u^T\tilde{A}^T\tilde{A}u. \quad (3.19)$$

If $\lambda < \tilde{\delta}$, then, because the right-hand side of (3.19) is negative, we must have $\lambda < 0$, and

so $\lambda_1 \geq \delta$. But by the implication drawn from (3.18), $\lambda_1 = \delta$ if and only if $\tilde{A}$ is rank deficient. This proves the last statement of the theorem.

If $\lambda > \tilde{\delta}$, we deduce from (3.19) that

$$\lambda\|u\|^2 \leq -\lambda_{\min}(H)\|u\|^2 + (\lambda - \tilde{\delta})^{-1}\sigma_{\max}(\tilde{A})^2\|u\|^2.$$

Upon simplifying and substituting ℓ for λ, we see that the quadratic

$$\ell^2 + (\lambda_{\min}(H) - \tilde{\delta})\ell - (\sigma_{\max}(B)^2 + \lambda_{\min}(H)\tilde{\delta})$$

in ℓ takes a nonpositive value when evaluated at an eigenvalue $\lambda > \delta$ of K. In particular this must be true of λ_m, which yields (3.17b).

If $\lambda < 0$, (3.19) yields the bound

$$\lambda\|u\|^2 \geq -\lambda_{\max}(H)\|u\|^2 + (\lambda - \max(1, \delta))^{-1}\sigma_{\max}(\tilde{A})^2\|u\|^2.$$

In turn, this implies that the quadratic in ℓ,

$$\ell^2 + (\lambda_{\max}(H) - \tilde{\delta})\ell - (\sigma_{\max}(B)^2 + \lambda_{\max}(H)\tilde{\delta}),$$

takes a nonpositive value when evaluated at an eigenvalue $\lambda < 0$ of K. In particular this must be true of $\lambda_n < 0$, which yields (3.17a). We now establish (3.17a). we have two cases for λ

1. If $\lambda < 0$ by multiplying u^T and v^T to left hand side of (3.18a) and (3.18a) respectively we have

$$-u^T H u + u^T \tilde{A}^T v = \lambda\|u\|^2, \tag{3.20}$$

$$v^T \tilde{A}^T u + v^T I_\delta v = \lambda\|v\|^2. \tag{3.21}$$

By subtracting (3.20) from (3.21) we obtain

$$-u^T H u - v^T I_\delta v = \lambda(\|u\|^2 - \|v\|^2). \tag{3.22}$$

Since we have

$$-u^T H u \leq -\lambda_{\min}(H)\|u\|^2 \leq -\rho\|u\|^2,$$

$$-v^T I_\delta v \leq -\tilde{\delta}\|v\|^2,$$

from (3.22) we get

$$\lambda\|u\|^2 - \lambda\|v\|^2 \leq -\rho\|u\|^2 - \tilde{\delta}\|v\|^2.$$

Therefore, the following inequality is obtained

$$(\lambda + \rho)\|u\|^2 \leq (\lambda - \tilde{\delta})\|v\|^2 \leq 0.$$

So necessarily u must be nonzero and we have

$$\lambda_{-1} \leq -\rho.$$

2. If $\lambda > 0$ using $u = (H + \lambda I)^{-1}\tilde{A}^T v$ from (3.20) and substituting in (3.21) we get

$$\tilde{A}\left(H + \lambda I\right)^{-1}\tilde{A}^T v + I_\delta v = \lambda v,$$

by multiplying v^T from left we have that

$$v^T\tilde{A}\left(H + \lambda I\right)^{-1}\tilde{A}^T v + v^T I_\delta v = \lambda\|v\|^2. \tag{3.23}$$

Since

$$\begin{aligned}
&v^T\tilde{A}\left(H + \lambda I\right)^{-1}\tilde{A}^T v \geq \\
&\lambda_{\min}\left(\tilde{A}(H + \lambda I)^{-1}\tilde{A}^T\right)\|v\|^2 \geq \\
&\lambda_{\min}\left(\tilde{A}\tilde{A}^T\right)\lambda_{\min}\left((H + \lambda I)^{-1}\right)\|v\|^2 \geq \\
&\sigma_{\min}(\tilde{A})^2\frac{1}{\lambda_{\max}(H) + \lambda}\|v\|^2
\end{aligned}$$

and $v^T I_\delta v \geq \tilde{\delta}\|v\|^2$ from (3.23) we have that

$$\lambda\|v\|^2 \geq \left(\frac{\sigma_{\min}(\tilde{A})^2}{\lambda_{\max}(H) + \lambda} + \tilde{\delta}\right)\|v\|^2 \geq \tilde{\delta}\|v\|^2$$

So we have

$$\lambda > \min(1, \delta),$$

and

$$\lambda_1 > \tilde{\delta},$$

which completes the proof.

□

Corollary 3.1.3. *For all $(x, z) > 0$ and $\rho > 0$ and $\delta > 0$ the smallest positive and the largest negative eigenvalues of K satisfy*

1. $\lambda_1 > \tilde{\delta}$
2. $\lambda_{-1} \leq -\rho$
3. $\|K\|^{-1} \leq \frac{1}{\min(\rho,\tilde{\delta})}$

Proof. Because $\|K\|^{-1} = \max(\frac{-1}{\lambda_{-1}}, \frac{1}{\lambda_1})$,

$$\|K\|^{-1} \leq \max\left(\frac{1}{\rho}, \frac{1}{\tilde{\delta}}\right) = \frac{1}{\min(\rho, \tilde{\delta})}.$$

□

Turning now to the right hand side of the Newton system (3.6), i.e.,

$$-Ax - r + d, \tag{3.24a}$$
$$-c + A^T r + B^T y - \rho s + z, \tag{3.24b}$$
$$-\rho x + \rho(s + x_k), \tag{3.24c}$$
$$-\delta y + \delta(w + y_k), \tag{3.24d}$$
$$b - Bx - \delta w, \tag{3.24e}$$
$$XZ - \sigma\mu e, \tag{3.24f}$$

from (3.24a) we have

$$-A(x + \alpha\Delta x) - (r + \alpha\Delta r) + d = -Ax - r + d - \alpha(A\Delta x + \Delta r).$$

Since

$$\Delta r = -A\Delta x - Ax - r + d, \tag{3.25}$$

(3.25) becomes

$$-Ax - r + d - \alpha(-Ax - r + d) = (1 - \alpha)(Ax - r + d). \tag{3.26}$$

From (3.24b) we have

$$\begin{aligned}
&-c + A^T(r + \alpha\Delta r) + B^T(y + \alpha\Delta y) - \rho(s + \alpha\Delta s) + z + \alpha\Delta z = \\
&-c + A^T r + \alpha A^T\Delta r + B^T y + \alpha B^T\Delta y - \rho s - \alpha\rho\Delta s + z + \alpha\Delta z = \\
&(-c + A^T r + B^T y - \rho s + z) - \alpha[-A^T\Delta r - B^T\Delta y + \rho\Delta s + \\
&(-\rho\Delta s + A^T\Delta r + B^T\Delta y - c + A^T r + B^T y + z)],
\end{aligned}$$

the last equality becomes

$$-\alpha(-c + A^T r + B^T y - \rho s + z)(1 - \alpha)(-c + A^T r + B^T y - \rho s + z). \tag{3.27}$$

From (3.24c) we have

$$\rho(s + \alpha\Delta s) = \rho s + \rho\alpha\Delta x - \rho\alpha s = (1 - \alpha)\rho s + \rho\alpha\Delta x, \tag{3.28}$$

and from (3.24d) we have

$$\delta(w + \alpha\Delta w) = \delta w + \delta\alpha\Delta w = \delta w + \delta\alpha(\Delta y - w) = (1 - \alpha)\delta w + \delta\alpha\Delta y. \tag{3.29}$$

From (3.24e) and $\delta\Delta w = -B\Delta x + b - Bx - \delta w$ we arrive at

$$b - B(x + \alpha\Delta x) - \delta(w + \alpha\Delta w) = b - Bx - \alpha B\Delta x - \delta w - \alpha\delta\Delta w = (1 - \alpha)(b - Bx - \delta w). \tag{3.30}$$

From (3.24f) and the fact that $Z\Delta x + X\Delta z = -Xz + \sigma\mu e$ we have

$$z^T\Delta x + x^T\Delta z = -x^T z + n\sigma\mu = -x^T z + \sigma x^T z = -(1 - \sigma)x^T z, \tag{3.31}$$

and

$$z_i[\Delta x]_i + x_i[\Delta z]_i = -x_i z_i + \sigma\frac{x^T z}{n}. \tag{3.32}$$

3.1.1 Fixed Regularization

Our first method, described in Algorithm 3.1.1, holds the regularization parameters ρ and δ fixed at all iterations and enforces conditions (3.12a)-(3.12d) at each iteration. Since the regularization parameters are constant in this section, we simply denote them ρ and δ for readability. Convergence properties rely on the following technical lemma.

Algorithm 3.1.1 Variation of the primal–dual method with constant regularization

Apply Algorithm 3.0.1 with $\rho_k := \rho_0 > 0$ and $\delta_k := \delta_0 > 0$ for all k. In step 3, only conditions (3.12a)- (3.12d) are enforced.

Lemma 3.1.4. *Suppose that $(\Delta x, \Delta r, \Delta y, \Delta z)$ is given by Step 2 of Algorithm 3.1.1, and the sequence $\{(s_k, w_k, z_k)\}$ is bounded. Then there exists a constant π, dependent only on n, such that*

$$|[\Delta x]_i[\Delta z]_i| \leq \pi, \quad |[\Delta x]_i[\Delta z]_i - \gamma_C \Delta x^T \Delta z/n| \leq \pi, \quad \textit{and}$$

$$|[\Delta x]_i[\Delta z]_i - \bar{\gamma}_C \Delta x^T \Delta z/n| \leq \pi.$$

Proof. The proof is similar to (Friedlander and Orban, 2012, Lemma 5.3). In order to prove the required result, it is sufficient to demonstrate that $(\Delta x, \Delta r, \Delta y, \Delta z)$ is bounded. To that end, we first show that $(\Delta x, \Delta r, \Delta y)$ is bounded, and second show that Δz is bounded. We have from (3.12b) and (3.13) that

$$\|Bx_k - b\| \leq \|Bx_k + \delta w_k - b\| + \delta\|w_k\| \leq x^T z_k/\gamma_P + \delta \sup_k \|w_k\|, \leq x_0^T z_0/\gamma_P + \delta \sup_k \|w_k\|$$

$$\|d - Ax_k - r_k\| \leq x_k^T z_k/\gamma_R \leq x_0^T z_0/\gamma_R,$$

which shows that the second block of the right-hand side in (3.9) is bounded. We now show that the first block in (3.9) is bounded. It follows from (3.12a) that

$$\frac{\sigma}{\bar{\gamma}_C} z \leq \sigma\mu X^{-1} e \leq \frac{\sigma}{\gamma_C} z,$$

componentwise. As a consequence,

$$\|\sigma_k \mu_k X^{-1} e - z_k\| \leq M \sup_k \|z_k\|, \quad \text{where} \quad M := \max\left(\left| \frac{\sigma}{\gamma_C} - 1 \right|, \left| \frac{\sigma}{\bar{\gamma}_C} - 1 \right| \right).$$

Combining this last inequality with (3.12d) and (3.13), we obtain

$$\|c - A^T r_k - B^T y_k - \sigma\mu X^{-1} e\| = \|c - A^T r_k - B^T y_k - \sigma\mu X^{-1} e \pm \rho s_k \pm z_k\|$$

$$\leq x_0^T z_0^T/\gamma_D + \rho \sup_k \|s_k\| + M \sup_k \|z_k\|.$$

By Corollary 3.1.3, the inverse of the matrix in (3.9) is bounded, and so $(\Delta x, \Delta r, \Delta y)$ is

bounded because the right-hand side of (3.9) is bounded. To show that Δz is bounded, we note that Δz satisfies (3.9), which has a bounded right-hand side. □

Our convergence analysis rests upon a variation of (Friedlander and Orban, 2012, Theorem 5.4) stating the convergence properties of Algorithm 3.1.1. The next result implies that the duality measure μ_k converges to zero under a boundedness assumption.

Theorem 3.1.5. *Suppose that Algorithm 3.1.1 with $\epsilon = 0$ generates the sequence $\{v_k\}$, and that the sequence $\{(s_k, w_k, z_k)\}$ is bounded. Then $\mu_k \to 0$.*

Proof. We follow Kojima et al. (1993), and express (3.13) in Step 3 of the algorithm as

$$(f_i(\alpha)), \bar{f}_i(\alpha) \geq 0, \quad h(\alpha) \geq 0, \quad g_P(\alpha) \geq 0, \quad g_R(\alpha) \geq 0, \quad \textit{and} \quad g_D(\alpha) \geq 0, \tag{3.33}$$

for $i = 1, \cdots, n$, where, dropping for the moment the subscript k,

$$f_i(\alpha) := ([x]_i + \alpha[\Delta x]_i)^T([z]_i + \alpha[\Delta z]_i) - \gamma_C(x + \alpha\Delta x)^T(z + \alpha\Delta z)/n, \tag{3.34a}$$
$$\bar{f}_i(\alpha) := \bar{\gamma}_C(x + \alpha\Delta x)^T(z + \alpha\Delta z)/n - ([x]_i + \alpha[\Delta x]_i)^T([z]_i + \alpha[\Delta z]_i), \tag{3.34b}$$
$$h(\alpha) := (1 - \alpha(1 - \beta))x^T z - (x + \alpha\Delta x)^T(z + \alpha\Delta z), \tag{3.34c}$$
$$g_P(\alpha) := (x + \alpha\Delta x)^T(z + \alpha\Delta z) - \gamma_P\|B(x + \alpha\Delta x) + \delta(w + \alpha\Delta w) - b\|, \tag{3.34d}$$
$$g_R(\alpha) := (x + \alpha\Delta x)^T(z + \alpha\Delta z) - \gamma_R\|A(x + \alpha\Delta x) + (r + \alpha\Delta r - d)\|, \tag{3.34e}$$
$$g_D(\alpha) := (x + \alpha\Delta x)^T(z + \alpha\Delta z) - \gamma_D\|c + \rho(s + \alpha\Delta s) - B^T(y + \alpha\Delta y) \tag{3.34f}$$
$$- A^T(r + \alpha\Delta r) - (z + \alpha\Delta z)\|, \tag{3.34g}$$

and by contradiction, we assume that $x^T z \geq \bar{\epsilon}$ for some $\bar{\epsilon} > 0$. We use (3.31) and (3.32) and Lemma 3.1.4, and the inequality $x^T z \geq \bar{\epsilon}$ to show that $f_i(\alpha) \geq 0$, $\bar{f}_i(\alpha) \geq 0$ and $h(\alpha) \geq 0$. We establish a similar property for $g_P(\alpha)$, $g_R(\alpha)$, and $g_D(\alpha)$. Using (3.30), (3.27), (3.31), and (3.34a) and the rest of proof is similar to (Friedlander and Orban, 2012, Theorem 5.4). □

In the next chapter, we propose a variation of Algorithm 3.0.1 in which the linear system (3.9) differs. Partial block elimination is not applied and this results in a larger but sparser SQD system. The convergence analysis rests upon a variation on (Armand and Benoist, 2011, Theorem 1) stating that the inverse of the coefficient matrix of Newton system remains uniformly bounded as long as $\{x_k\}$ and $\{z_k\}$ remain bounded away from zero.

CHAPTER 4

A REGULARIZED INTERIOR-POINT METHOD FOR CONSTRAINED LINEAR LEAST SQUARES

4.1 Background and Preliminaries

As a convex quadratic program, the dual of (1.1) may be written as the constrained linear least-squares problem

$$\begin{aligned} \underset{x,y,z}{\text{maximize}} \quad & b^T y - (A^T d)^T x - \tfrac{1}{2}\|Ax - d\|^2 \\ \text{subject to} \quad & B^T y + z - A^T A x = c - A^T d, \ z \geq 0, \end{aligned} \tag{4.1}$$

where $y \in \mathbb{R}^m$ and $z \in \mathbb{R}^n$ are the vectors of Lagrange multipliers associated to the equality constraints and bounds of (1.1), respectively.

Friedlander and Orban (2012) justify the regularization (1.2) of (1.1) as an application of the proximal method of multipliers of Rockafellar (1976). It consists in the addition of a proximal-point term $\frac{1}{2}\rho\|x - x_k\|^2$, to which we will refer as a *primal regularization* term, and of augmented Lagrangian terms consisting of the objective term $\frac{1}{2}\delta\|w + y_k\|^2$ and the constraint residual term δw, to which we will collectively refer as a *dual regularization* term. The regularized problem (1.2) is still a convex quadratic program and its dual may be written as the regularized constrained linear least-squares problem

$$\begin{aligned} \underset{x,s,y,z}{\text{maximize}} \quad & b^T y - (A^T d)^T x - \tfrac{1}{2}\|Ax - d\|^2 - \tfrac{1}{2}\delta\|y - y_k\|^2 - \tfrac{1}{2}\rho\|s + x_k\|^2 \\ \text{subject to} \quad & B^T y + z - A^T A x - \rho s = c - A^T d, \ z \geq 0, \end{aligned} \tag{4.2}$$

where $s = x - x_k$ are auxiliary variables playing the same role as w in (1.2). The strength of this regularization approach is that (4.2) is precisely the primal-dual regularization of (4.1).

For convenience, we let $u := (x, s, w, y, z)$ and we define the function

$$F_k(u; \rho, \delta, \tau) := \begin{bmatrix} c - A^T d - B^T y - z + A^T A x + \rho s \\ \rho x - \rho(s + x_k) \\ \delta y - \delta(w + y_k) \\ Bx + \delta w - b \\ Xz - \tau e \end{bmatrix}, \tag{4.3}$$

where $\tau \geq 0$ is a parameter.

Using this notation, the common necessary and sufficient optimality conditions of (1.2) and (4.2) may be written compactly as

$$F_k(u; \rho, \delta, 0) = 0, \qquad (x, z) \geq 0.$$

Note also that setting additionally $\rho = \delta = 0$ recovers the optimality conditions of the original primal-dual pair (1.1) and (4.1).

An interior-point method applied to the primal-dual pair (1.2) and (4.2) iteratively seeks approximate solutions to the nonlinear system

$$F_k(u; \rho, \delta, \tau_k) = 0, \quad (x, z) > 0,$$

for a sequence of parameters $\{\tau_k\} \downarrow 0$. For each fixed value of τ_k, a Newton step Δu is computed from the current approximation u_k as the solution of the linear system

$$\nabla_u F_k(u_k; \rho, \delta, \tau_k) \Delta u = -F_k(u_k; \rho, \delta, \tau_k),$$

where the Jacobian is given by

$$\nabla_u F_k(u_k; \rho, \delta, \tau_k) = \begin{bmatrix} A^T A & \rho I & & -B^T & -I \\ \rho I & -\rho I & & & \\ & & -\delta I & \delta I & \\ B & & \delta I & & \\ Z & & & & X \end{bmatrix}.$$

Of particular concern is that the matrix $A^T A$ is (nearly) dense if A has a (nearly) dense row. One way to circumvent this difficulty is to introduce $\xi := d - A\Delta x - Ax$. An equivalent way to write the previous system is then

$$\begin{bmatrix} & A^T & \rho I & & -B^T & -I \\ A & I & & & & \\ & & & \vdots & & \end{bmatrix} \begin{bmatrix} \Delta x \\ \xi \\ \vdots \end{bmatrix} = \begin{bmatrix} -c + B^T y + z - \rho s \\ d - Ax \\ \vdots \end{bmatrix},$$

where the ellipses indicate that the rest of the system is unchanged. This system is larger but sparser and does away with the matrix-matrix product $A^T A$.

A more natural way to give rise to the previous sparse Jacobian is to systematically

transform every problem of the form (1.1) to the form

$$\begin{aligned} \underset{x,r}{\text{minimize}} \quad & c^T x + \tfrac{1}{2}\|r\|^2 \\ \text{subject to} \quad & Bx = b, \quad Ax + r = d, \quad x \geq 0, \end{aligned} \tag{4.4}$$

whose dual may be written

$$\begin{aligned} \underset{x,r,y,z}{\text{maximize}} \quad & b^T y - (A^T d)^T x - \tfrac{1}{2}\|r\|^2 \\ \text{subject to} \quad & B^T y + z + A^T r = c, \quad Ax + r = d, \quad z \geq 0. \end{aligned} \tag{4.5}$$

It is interesting to note that the constraints $Ax + r = d$, defining the residual r, appear in both the primal and the dual problem. This simple fact turns out to guide our choice of regularization.

We formulate the regularization of (4.4) as

$$\begin{aligned} \underset{x,r,w}{\text{minimize}} \quad & c^T x + \tfrac{1}{2}\|r\|^2 + \tfrac{1}{2}\rho\|x - x_k\|^2 + \tfrac{1}{2}\delta\|w + y_k\|^2 \\ \text{subject to} \quad & Bx + \delta w = b, \quad Ax + r = d, \quad x \geq 0. \end{aligned} \tag{4.6}$$

The former regularization differs from (1.2) in two respects. Firstly, no primal regularization term is added for the variables r because the objective function of (4.4) is already strictly convex in r. Secondly, the constraints $Ax + r = d$ are not regularized since they already have full row rank. Since the variables r do not appear elsewhere in the constraints, the equality constraints of (4.6) have full row rank. No harm would be done in regularizing $Ax + r = d$ although a larger system would be obtained. The dual of (4.6) may be stated as

$$\begin{aligned} \underset{x,r,y,s}{\text{maximize}} \quad & b^T y - (A^T d)^T x - \tfrac{1}{2}\|r\|^2 - \tfrac{1}{2}\delta\|y - y_k\|^2 - \tfrac{1}{2}\rho\|s + x_k\|^2 \\ \text{subject to} \quad & A^T r + B^T y + z - \rho s = c, \\ & Ax + r = d, \quad z \geq 0, \end{aligned} \tag{4.7}$$

where we introduced the auxiliary variables $s = x - x_k$. Because the dual (4.7) also features the full-rank constraints $Ax + r = d$, it offers an additional elegant justification for omitting the primal regularization term for the variables r in (4.6). Indeed, regularizing those constraints in (4.7) precisely amounts to adding the primal regularization term in question. It is now visible that upon setting $\rho = \delta = 0$, (4.6) and (4.7) coincide with (4.4) and (4.5), respectively. In the rest of this document, we concentrate on the formulation (4.6)–(4.7).

Proceeding as before, we let $v := (x, r, s, w, y, z)$ and define

$$\Psi_k(v; \rho, \delta, \tau) := \begin{bmatrix} c + \rho s - B^T y - A^T r - z \\ \rho x - \rho(s + x_k) \\ \delta y - \delta(w + y_k) \\ Bx + \delta w - b \\ Ax + r - d \\ Xz - \tau e \end{bmatrix}. \tag{4.8}$$

Note that the definition of Ψ does not involve the Lagrange multipliers associated to the constraints $Ax + r = d$. This is because those can be readily eliminated and are always equal to r. Once again, the optimality conditions of (4.6)–(4.7) can be succinctly stated as $\Psi_k(v; \rho, \delta, 0) = 0$ and $(x, z) \geq 0$ while those of (4.4)–(4.5) can be expressed as $\Psi_k(v; 0, 0, 0) = 0$ and $(x, z) \geq 0$.

In the next section, we outline the main features of a long-step interior-point method applied to (4.6)–(4.7).

4.2 Interior-Point Method

This section describes the linear systems to be solved at each iteration of an interior-point method applied to (4.6)–(4.7) and the neighborhood of the central path used to guide the iterates to a solution of (1.1) and (4.1). We end the section by stating our algorithm formally.

4.2.1 Linear Systems

As in the previous section, the Newton correction Δv for (4.8) from the current approximation v_k with barrier parameter τ_k solves the system $\nabla_v \Psi_k(v_k; \rho, \delta, \tau_k) \Delta v = -\Psi_k(v_k; \rho, \delta, \tau_k)$. After eliminating Δs and Δw, and slightly rearranging, there remains

$$\begin{bmatrix} -\rho I & A^T & B^T & I \\ A & I & & \\ B & & \delta I & \\ Z_k & & & X_k \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} c - B^T y_k - A^T r_k - z_k \\ d - A x_k - r_k \\ b - B x_k \\ \tau_k e - X_k z_k \end{bmatrix}. \tag{4.9}$$

The remaining directions may be recovered via

$$\Delta w = \Delta y - w_k, \qquad \Delta s = \Delta x - s_k. \tag{4.10}$$

Note that upon setting $\rho = \delta = 0$, we recover the Newton equations used to compute a step from the k-th iterate of an interior-point method applied to (4.4)–(4.5).

Rather than using (4.9) directly, our implementation, described in §4.4 makes use of the following symmetrization, obtained via the similarity transformation defined by the diagonal matrix blkdiag$(I, I, I, Z_k^{-\frac{1}{2}})$:

$$\begin{bmatrix} -\rho I & A^T & B^T & Z_k^{\frac{1}{2}} \\ A & I & & \\ B & & \delta I & \\ Z_k^{\frac{1}{2}} & & & X_k \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta y \\ Z_k^{-\frac{1}{2}} \Delta z \end{bmatrix} = \begin{bmatrix} c - B^T y_k - A^T \lambda_k - z_k \\ d - A x_k - r_k \\ b - B x_k \\ \tau_k Z_k^{-\frac{1}{2}} e - X_k Z_k^{\frac{1}{2}} e \end{bmatrix}. \tag{4.11}$$

The above symmetric system differs from that traditionally used in interior-point methods, which results from an additional step of block Gaussian elimination about the $(4, 4)$ block. Our motivation for using (4.11) stems from recent results of Greif et al. (2012) who establish that as long as ρ and δ remain bounded away from zero and strict complementarity holds at the limiting solution, the above coefficient matrix remains uniformly bounded and uniformly nonsingular. Moreover, its condition number remains sufficiently small along the iterations that a reasonable number of significant digits in the solution may be expected. By contrast, the coefficient matrix of the traditional system is increasingly ill-conditioned as $\tau_k \downarrow 0$ and typically diverges, even if strict complementarity holds.

Note that the coefficient matrix of (4.11) is symmetric and quasi definite (Vanderbei, 1995). It is therefore strongly factorizable, i.e., any symmetric permutation of it possesses a LDL^T factorization with L unit lower triangular and D diagonal indefinite. The computation of this factorization is typically cheaper than that of a sparse symmetric indefinite factorization since pivoting need only be concerned with sparsity (Gill et al., 1996).

There is an elegant interpretation of (4.11) that is particularly fitting in the present least-square framework. For simplicity of exposition, let us rewrite (4.11) as

$$\begin{bmatrix} \rho I & B^T \\ B & -D \end{bmatrix} \begin{bmatrix} \Delta x \\ t \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$

where we defined

$$B^T = \begin{bmatrix} A^T & B^T & Z_k^{\frac{1}{2}} \end{bmatrix}, \quad D = \text{blkdiag}(I, \delta I, X_k), \quad t = (\Delta r, \Delta y, Z_k^{-\frac{1}{2}} \Delta z)$$

and the right-hand side is defined accordingly. This last system may be solved in two stages. Firstly, let $\bar{t} := -D^{-1} g$. Since D is diagonal, computing $\bar{t}$ is trivial. The system may now

equivalently be written

$$\begin{bmatrix} \rho I & B^T \\ B & -D \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta t \end{bmatrix} = \begin{bmatrix} \bar{f} \\ 0 \end{bmatrix}$$

where $t = \bar{t} + \Delta t$ and $\bar{f} := f - B^T \bar{t}$. This shifted system represents the necessary and sufficient optimality conditions of the unconstrained regularized linear least-squares problem

$$\underset{\Delta t}{\text{minimize}} \ \tfrac{1}{2}\|B^T \Delta t - \bar{f}\|^2_{M^{-1}} + \tfrac{1}{2}\|\Delta t\|^2_D, \tag{4.12}$$

where $M := \rho I$.

Any method requiring the solution of a symmetric quasi-definite system at each iteration may be interpreted as solving a regularized linear least-squares problem of the form (4.12) at each iteration. In the present context of solving (1.1), this interpretation is particularly fitting. It also forms the basis for the iterative methods developed by Arioli and Orban (2012) and hence paves the way to a matrix-free interior-point method for (1.1). In §4.4 however, we solve (4.11) using a sparse LDL^T factorization.

4.2.2 Neighborhood of the Central Path

The *central path* is the set of exact roots $v(\tau)$ of $\Psi_k(v; \rho, \delta, \tau)$ for $\tau > 0$. For fixed ρ and δ, as τ approaches zero, it can be shown that $v(\tau)$ approaches a solution of (4.4)–(4.5). Note that the last block equation of $\Psi_k(v; \rho, \delta, \tau) = 0$ implies that $[x(\tau)]_i[z(\tau)]_i = \tau$ for all $i = 1, \ldots, n$.

Typical interior-point methods compute estimates v_k for some sequence $\{\tau_k\} \downarrow 0$ that are close, in some sense, to $v(\tau_k)$. This concept of proximity is formalized by a neighborhood of the central path. A usual choice is $\tau_k := \sigma_k \mu_k$, where $\sigma_k \in (0, 1)$ is a *centering* parameter and $\mu_k := x_k^T z_k / n$, which measures the average centrality. If v_k lies exactly on the central path, we have $[x_k]_i[z_k]_i = \mu_k$ for all $i = 1, \ldots, n$. The quantity μ_k is also directly proportional to the duality gap between (4.4) and (4.5) in the case where v_k is primal-dual feasible.

A difference between our approach and traditional interior-point methods is that during the course of the iterations, the regularization parameters ρ and δ may be updated. At the k-th iteration, the current iterate is v_k and the regularization parameters have values ρ_k and

δ_k. The neighborhood $\mathcal{N}_k$ is defined by an appropriate *subset* of the following conditions:

$$\bar{\gamma}_C x^T z/n \geq [x]_i[z]_i \geq \gamma_C x^T z/n, \tag{4.13a}$$

$$x^T z \geq \gamma_P \|Bx + \delta_k w - b\|, \tag{4.13b}$$

$$x^T z \geq \gamma_R \|Ax + r - d\|, \tag{4.13c}$$

$$x^T z \geq \gamma_D \|c + \rho_k s - B^T y - A^T r - z\|, \tag{4.13d}$$

$$x^T z \geq \gamma_S \|\rho_k x - \rho_k(s + x_k)\|, \tag{4.13e}$$

$$x^T z \geq \gamma_W \|\delta_k y - \delta_k(w + y_k)\|, \tag{4.13f}$$

where $0 < \gamma_C < 1 < \bar{\gamma}_C$ and $(\gamma_P, \gamma_R, \gamma_D, \gamma_S, \gamma_W) > 0$ are given constants. Our interior-point scheme computes a steplength $\alpha_k \in (0, 1]$ as well as updated regularization parameters ρ_{k+1} and δ_{k+1} so that the next iterate $v_{k+1} = v_k(\alpha_k) := v_k + \alpha_k \Delta v \in \mathcal{N}_{k+1}$.

4.2.3 Algorithm

Our algorithm is the same as (Friedlander and Orban, 2012, Algorithm 4.1) with the exception of the linear system used in Step 2, and is formalized as Algorithm 4.2.1.

Algorithm 4.2.1 Primal-Dual Regularized Interior-Point Algorithm

Step 0 [**Initialize**] Choose minimum and maximum centering parameters $0 < \sigma_{\min} \leq \sigma_{\max} < 1$, a constant $\sigma_{max} < \beta < 1$, proximity parameters $0 < \gamma_C < 1 < \bar{\gamma}_C$ and $(\gamma_P, \gamma_R, \gamma_D, \gamma_S, \gamma_W) > 0$, initial regularization parameters $\rho_0 > 0$, $\delta_0 > 0$, and a stopping tolerance $\epsilon > 0$. Let the neighborhood of the central path be defined by (4.13a), (4.13b), (4.13c) and (4.13d). Choose initial primal $x_0 \in \mathbb{R}^n_{++}$, $r_0 \in \mathbb{R}^m$, $w_0 \in \mathbb{R}^m$ and dual guesses $s_0 \in \mathbb{R}^n$, $y_0 \in \mathbb{R}^m$, and $z_0 \in \mathbb{R}^n_{++}$ so that $v_0 \in \mathcal{N}_0$. Set $\mu_0 := x_0^T z_0/n$ and $k = 0$.

Step 1 [**Test convergence**] If $x_k^T z_k \leq \epsilon$, declare convergence.

Step 2 [**Step computation**] Choose a centering parameter $\sigma_k \in [\sigma_{\min}, \sigma_{\max}]$. Compute the Newton step Δv_k from v_k, e.g., by solving (4.9) with $\tau_k := \sigma_k \mu_k$ and recovering the remaining components from (4.10).

Step 3 [**Linesearch**] Select $\delta_{k+1} \in (0, \delta_k]$ and $\rho_{k+1} \in (0, \rho_k]$ and compute α_k as the largest $\alpha \in (0, 1]$ such that

$$v_k(\alpha) \in \mathcal{N}_{k+1} \quad \text{and} \quad \mu_k(\alpha) \leq (1 - \alpha(1 - \beta))\mu_k,$$

where $\mu_k(\alpha) := x_k(\alpha)^T z_k(\alpha)/n$.

Step 4 [**Update iterates**] Set $v_{k+1} := v_k(\alpha_k)$, $\mu_{k+1} := \mu_k(\alpha_k)$. Increment k by 1 and go to Step 1.

Friedlander and Orban (2012) present two variants of Algorithm 4.2.1. The first variant keeps ρ and δ fixed throughout the iterations and only considers (4.13a), (4.13b), (4.13c) and (4.13d) in the definition of $\mathcal{N}_k$. In the second variant, ρ_k and δ_k are allowed to decrease at most linearly and $\mathcal{N}_k$ is defined by (4.13a), (4.13e) and (4.13f). Both variants have similar convergence properties. In our implementation, described in §4.4, we initially decrease ρ_k and δ_k so as to speed up convergence and eventually keep them fixed at a level that guarantees numerical stability of the factorization of the coefficient matrix of (4.11). For this reason, in the next section, we only cover the convergence properties of the variant with fixed regularization parameters.

4.3 Convergence Analysis

Our convergence analysis rests upon a variation on (Armand and Benoist, 2011, Theorem 1) stating that the inverse of the coefficient matrix of (4.9) remains uniformly bounded as long as $\{x_k\}$ and $\{z_k\}$ remain bounded away from zero. The reason for this unconventional last assumption is that the convergence proof proceeds by contradiction on the fact that $\{\mu_k\}$ converges to zero. We begin by stating the result on the boundedness of the inverse coefficient matrix.

Theorem 4.3.1. *Let $\{M_k\}$ be a sequence of $n \times n$ real symmetric matrices, $\{A_k\}$ be a sequence of $p \times n$ real matrices, $\{B_k\}$ be a sequence of $m \times n$ real matrices, and $\{\delta_k\}$ be a sequence of positive numbers. Let $\{x_k\}$ and $\{z_k\}$ be two sequences of $\mathbb{R}^n$ with positive components. Define for all $k \in \mathbb{N}$,*

$$J_k := \begin{bmatrix} M_k & A_k^T & B_k^T & -I \\ A_k & -I & & \\ B_k & & -\delta_k I & \\ Z_k & & & X_k \end{bmatrix}.$$

Assume the following properties are satisfied:

1. *The sequences $\{M_k\}$, $\{A_k\}$ and $\{B_k\}$ are bounded.*
2. *The sequence $\{\delta_k\}$ is bounded away from zero.*
3. *There exists $\eta > 0$ such that for all $k \in \mathbb{N}$ and all $i \in \{1, \ldots, n\}$,*

$$[x_k]_i [z_k]_i \geq \eta. \tag{4.14}$$

4. *There exists $\lambda > 0$ such that for all $k \in \mathbb{N}$ and all $d \in \mathbb{R}^n$,*

$$d^T H_k d \geq \lambda \|d\|^2,$$

where $H_k := M_k + X_k^{-1} Z_k + \delta_k^{-1} B_k^T B_k + A_k^T A_k$.

Then the sequence $\{J_k^{-1}\}$ is well defined and bounded.

Proof. It suffices to apply (Armand and Benoist, 2011, Theorem 1) to the matrix $\operatorname{blkdiag}(I, \sqrt{\delta_k} I, I, I) J_k \operatorname{blkdiag}(I, \sqrt{\delta_k} I, I, I)$. □

Our interest in Theorem 4.3.1 is to set $M_k := \rho I$, $A_k := -A$ and $B_k := -B$ for all k. This guarantees that Assumption (1) is satisfied. Assumption (2) is satisfied since Algorithm 4.2.1 works with fixed regularization parameters. Assumption (4) is also satisifed because of our definition of M_k. Assumption (3) will be the main contradiction assumption. Note also that the matrix J_k of Theorem 4.3.1 is the coefficient matrix of (4.9) multiplied by the diagonal matrix $\operatorname{blkdiag}(-I, -I, -I, I)$. Therefore, their inverses are simultaneously uniformly bounded.

Our first result consists in giving conditions under which the right-hand side of (4.9) is bounded.

Lemma 4.3.2. *Let $\{v_k\}$ be the sequence generated by Algorithm 4.2.1. Assume $\{(s_k, w_k)\}$ remains bounded. Then the right-hand side of* (4.9) *is uniformly bounded.*

Proof. Using our assumption that $\{w_k\}$ is bounded, we have

$$\|b - Bx_k\| \leq \|b - Bx_k - \delta w_k\| + \delta\|w_k\| \leq \gamma_P^{-1} n\mu_k + \delta \sup_k \|w_k\|,$$

where we used the definition of $\mu_k = x_k^T z_k/n$ and (4.13b). Since Algorithm 4.2.1 ensures that $\{\mu_k\}$ is decreasing, the above establishes boundedness of $\{b - Bx_k\}$. The boundedness of $\{c - A^T r_k - B^T y_k - z_k\}$ follows similarly from the boundedness of $\{s_k\}$ and (4.13d). Boundedness of $\{Ax_k + r_k - d\}$ follows directly from (4.13c). Finally, Boundedness of $\{\sigma_k\mu_k - X_k z_k\}$ follows from (4.13a) and the boundedness of $\{\sigma_k\}$ and $\{\mu_k\}$. □

Using Theorem 4.3.1 and Lemma 4.3.2 we obtain uniform boundedness of the direction Δv under the contradiction assumption.

Lemma 4.3.3. *Let $\{v_k\}$ be the sequence generated by Algorithm 4.2.1. Assume $\{(s_k, w_k)\}$ remains bounded and assume that there exists $\eta > 0$ such that* (4.14) *is satisfied. Then the direction Δv is uniformly bounded.*

Proof. Using Theorem 4.3.1 and Lemma 4.3.2, we obtain that $(\Delta x, \Delta r, \Delta y, \Delta z)$ is uniformly bounded. Finally, (4.10) and boundedness of $\{s_k\}$ and $\{w_k\}$ yield boundedness of $\{\Delta s\}$ and $\{\Delta w\}$. □

A careful inspection of (Friedlander and Orban, 2012, §5.2 and §5.4) reveals that all that is required to establish that the sequence $\{\mu_k\}$ converges to zero in Algorithm 4.2.1 is precisely Lemma 4.3.3. More precisely, we have the following global convergence result.

Theorem 4.3.4 (Friedlander and Orban 2012, Theorem 5.10)**.** *Let $\{v_k\}$ be the sequence generated by Algorithm 4.2.1 with $\epsilon = 0$. Assume $\{(s_k, w_k)\}$ remains bounded. Then $\{\mu_k\} \to 0$.*

Proof. The proof is by contradiction. Assume that $\{\mu_k\} \not\to 0$. Then there must exist $\eta > 0$ such that (4.14) is satisfied. We conclude from Lemma 4.3.3 that Δv is uniformly bounded. The rest of the proof proceeds as (Friedlander and Orban, 2012, Theorem 5.4) to conclude that

$$0 < \eta \leq \mu_{k+1} \leq \gamma\mu_k \leq \cdots \leq \gamma^{k+1}\mu_0,$$

for some constant $\gamma \in (0, 1)$. Since the right-hand side of the previous inequalities converges to zero, we obtain the contradiction. □

The nature of the limit points of the sequence generated by Algorithm 4.2.1 is stated in the next two results. The first states that in general, a solution of a perturbed primal-dual pair is identified. This primal-dual pair coincides with the original pair (1.1) and (4.1) but has shifted linear terms and right-hand sides.

Theorem 4.3.5 (Friedlander and Orban 2012, Theorem 5.5)**.** *Let $\{v_k\}$ be the sequence generated by Algorithm 4.2.1 with $\epsilon = 0$. Assume $\{(s_k, w_k)\}$ remains bounded. If w_* and s_* denote particular limit points of $\{w_k\}$ and $\{s_k\}$ defined by subsequences indexed by the index set $\mathcal{K} \subseteq \mathbb{N}$, every limit point of $\{(x_k, r_k, z_k)\}_{\mathcal{K}}$ determines a primal-dual solution of the primal-dual pair*

$$\begin{aligned} \underset{x,r}{\text{minimize}} \quad & (c + \rho s_*)^T x + \tfrac{1}{2}\|r\|^2 \\ \text{subject to} \quad & Bx = b - \delta w_*, \quad Ax + r = d, \quad x \geq 0, \end{aligned} \tag{4.15}$$

and

$$\begin{aligned} \underset{x,r,y,z}{\text{maximize}} \quad & (b - \delta w_*)^T y - (A^T d)^T x - \tfrac{1}{2}\|r\|^2 \\ \text{subject to} \quad & B^T y + z + A^T r = c + \rho s_*, \quad Ax + r = d, \quad z \geq 0. \end{aligned} \tag{4.16}$$

It is now clear, in light of Theorem 4.3.5, that whenever $s_* = 0$ and $w_* = 0$, we recover a primal-dual solution of the original problem. That is the essence of the second result.

Theorem 4.3.6 (Friedlander and Orban 2012, Theorem 5.6)**.** *Let $\{v_k\}$ be the sequence generated by Algorithm 4.2.1 with $\epsilon = 0$. Assume $\{(s_k, w_k)\}$ remains bounded. Then*

1. *If $\{w_k\}_{\mathcal{K}} \to 0$ for some index set $\mathcal{K} \subseteq \mathbb{N}$, every limit point of $\{(x_k, r_k)\}_{\mathcal{K}}$ is feasible for* (1.1)*.*
2. *If $\{s_k\}_{\mathcal{K}'} \to 0$ for some index set $\mathcal{K}' \subseteq \mathbb{N}$, every limit point of $\{(x_k, r_k, z_k)\}_{\mathcal{K}'}$ determines a feasible point for* (4.1)*.*
3. *If $\{(s_k, w_k)\}_{\mathcal{K}''} \to 0$ for some index set $\mathcal{K}'' \subseteq \mathbb{N}$, every limit point of $\{(x_k, r_k, z_k)\}_{\mathcal{K}''}$ determines a primal-dual solution of* (1.1)–(4.1)*.*

4.4 Implementation and Numerical Results

Our implementation is strongly based on that of Friedlander and Orban (2012) with the difference that a step Δv is computed using (4.11) instead of a partial reduction of this system. The implementation accepts problems with free variables in so-called *slack form*

$$\underset{x,t}{\text{minimize}} \quad c^T x + \tfrac{1}{2}\|Ax - d\|^2 \quad \text{subject to } Bx + B_1 t = b,\ t \geq 0, \tag{4.17}$$

whose dual may be written

$$\begin{aligned} \underset{x,y,z}{\text{maximize}} \quad & b^T y - (A^T d)^T x - \tfrac{1}{2}\|Ax - d\|^2 \\ \text{subject to} \quad & B^T y - A^T A x = c - A^T d,\ B_1^T y + z = 0,\ z \geq 0. \end{aligned} \tag{4.18}$$

The problem is systematically turned to the form (4.4)–(4.5). The system used to compute a Newton step at iteration k now takes the form

$$\begin{bmatrix} -\rho I & & A^T & B^T & \\ & -\rho I & & B_1^T & Z^{\frac{1}{2}} \\ A & & I & & \\ B & B_1 & & \delta I & \\ & Z^{\frac{1}{2}} & & & T \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta t \\ \Delta r \\ \Delta y \\ Z^{-\frac{1}{2}} \Delta z \end{bmatrix} = \begin{bmatrix} c - A^T r - B^T y \\ -B_1^T y - z \\ d - Ax - r \\ b - Bx - B_1 t \\ Z^{-\frac{1}{2}}(\sigma_k \mu_k e - T_k z_k) \end{bmatrix},$$

where $T = \text{diag(t)}$. This system is solved by way of a sparse LDL^T factorization using MA57 from Duff (2004) as implemented in the HSL (2007) and setting the pivot tolerance to zero. Although in theory some components of T typically converge to zero, and therefore the limiting matrix is not symmetric and quasi definite, all components of T remain larger than, say, 10^{-8} in practice and we have not encountered numerical difficulties related to the factorization. In §4.5, we discuss alternatives that rule out such potential numerical difficulty. The important advantage of the coefficient matrix above is that its condition number remains uniformly bounded provided strict complementarity holds in the limit.

The algorithm implemented is a predictor-corrector variant of the long-step method described in Algorithm 4.2.1. All initialization and updates are as described by Friedlander and Orban (2012). In particular, ρ and δ are initialized to the value 1 and divided by 10 at each iteration but are not allowed to decrease below 10^{-8}. The method is implemented in the Python language as part of the NLPy library (Orban, 2012).

4.5 Discussion

Gill et al. (1996) show that the LDL^T factorization of a symmetric and quasi-definite matrix becomes increasingly unstable if either diagonal block approaches singularity or if an off-diagonal block becomes large. Specifically, we have the result

Theorem 4.5.1 (Gill et al. 1996, Result 4.2)**.** *Let*

$$K := \begin{bmatrix} H & B^T \\ B & -G \end{bmatrix}$$

be a symmetric quasi-definite matrix. The factorization $PKP^T = LDL^T$, where L is unit lower triangular and D is diagonal, is stable for every permutation matrix P if

$$\theta(K) := \left(\frac{\|B\|_2}{\max(\|G\|_2, \|H\|_2)} \right)^2 \max(\kappa_2(G), \kappa_2(H)),$$

is not too large, where κ_2 denotes the spectral condition number.

Clearly, if G approaches singularity, $\theta(K)$ becomes large. This is not to say that there does not exist some permutation P for which the factorization is stable. However, this permutation, if it exists, may not yield particularly sparse factors.

One possibility in this case is to resort to the usual symmetric indefinite factorization LBL^T, where B is now block diagonal. The additional cost incurred may be acceptable since it should only be necessary in the last few iterations. Another possibility is to perform the usual block elimination on (4.11) and reduce it to a system with coefficient matrix

$$\begin{bmatrix} -(X_k^{-1}Z_k + \rho I) & A^T & B^T \\ A & I & \\ B & & \delta I \end{bmatrix}.$$

Zero elements on the diagonal no longer occur but unfortunately the above matrix no longer has a bounded condition number. Theorem 4.5.1 also suggests that the LDL^T factorization is still unstable. A third possibility is to perform an additional transform by multiplying the coefficient matrix of (4.11) on the left and right by blkdiag$(I, I, I, X^{-\frac{1}{2}})$ and scale the vector

of unknowns and the right-hand side accordingly. This yields the coefficient matrix

$$\begin{bmatrix} -\rho I & A^T & B^T & X_k^{-\frac{1}{2}} Z_k^{\frac{1}{2}} \\ A & I & & \\ B & & \delta I & \\ X_k^{-\frac{1}{2}} Z_k^{\frac{1}{2}} & & & I \end{bmatrix}.$$

This time it is $\|B\|_2$ that becomes large in Theorem 4.5.1. Finally, eliminating the small diagonal elements of x in the vein of Gould (1986) again produces a limiting matrix that is not quasi definite. It appears difficult to maintain safe quasi definiteness in the limit if at least one bound constraint is active at a solution.

In our situation however, we conjecture that there exists a permutation that produces a stable factorization of (4.11) for the following reason. Suppose the sequences $\{x_k\}$ and $\{z_k\}$ generated by Algorithm 4.2.1 converge to x_* and z_*, respectively. Define the index sets

$$\mathcal{A} := \{i \mid [x_*]_i = 0\} \qquad \mathcal{I} := \{i \mid [x_*]_i > 0\}.$$

By complementarity, we have $[z_*]_i = 0$ for all $i \in \mathcal{I}$. If we assume that strict complementarity holds at (x_*, z_*), then we also have $[z_*]_i > 0$ for all $i \in \mathcal{A}$. Therefore, for all sufficiently large indices k, (4.13a) implies that

$$[x_k]_i = \Theta(\mu_k) \qquad [z_k]_i = \Theta(1) \quad (i \in \mathcal{A}) \tag{4.19}$$

$$[z_k]_i = \Theta(\mu_k) \qquad [x_k]_i = \Theta(1) \quad (i \in \mathcal{I}). \tag{4.20}$$

Consider the following example matrix representative of a problem with 3 variables and bound constraints only, two of which are active at a solution.

$$K = \begin{bmatrix} -1 & & & \sqrt{\mu} & & \\ & -1 & & & 1 & \\ & & -1 & & & 1 \\ \sqrt{\mu} & & & 1 & & \\ & 1 & & & \mu & \\ & & 1 & & & \mu \end{bmatrix}.$$

Without permutation, the LDL^T factorization of K yields

$$L = \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ -\sqrt{\mu} & & & 1 & & \\ & -1 & & & 1 & \\ & & -1 & & & 1 \end{bmatrix} \qquad D = \begin{bmatrix} -1 & & & & & \\ & -1 & & & & \\ & & -1 & & & \\ & & & 1+\mu & & \\ & & & & 1+\mu & \\ & & & & & 1+\mu \end{bmatrix}$$

Therefore, this factorization is stable in the sense that there exists a moderate constant $\gamma > 0$ such that

$$\| \, |L| \, |D| \, |L^T| \, \| \leq \gamma \|K\|,$$

where the absolute value of a matrix is the matrix of the absolute values of its elements—see Golub and Van Loan (1996). In fact, the relative error between the computed LDL^T and K is of the order of the machine epsilon for all μ larger than the machine epsilon. Note that μ never becomes that small in practice. If we exchange the first and second variables in K, the relative error for $\mu = 10^{-16}$ rises to about 70%. Though the above example is no proof, it leads us to speculate that as long as the small elements of X_k—i.e., those in $\mathcal{A}$—appear last on the diagonal, the factorization will be stable. This example is inspired by and related to similar examples by Vanderbei (1995) and Gill et al. (1996).

CHAPTER 5

APPLICATIONS

The practice of optimization depends not only on efficient and robust algorithms, but also on good modelling techniques, careful interpretation of results, and user-friendly software. Here, we provide three LS problems applications.

5.1 Constrained Curve Fitting

In the constrained curve fitting example with inconsistent inequality constraints, we fit the linear function

$$f(t) = x^T t,$$

where x, $t \in \mathbb{R}^n$, $\sum_{i=1}^{n} [t]_i = 1$ and $0 \leq [t]_i \leq 1$ for $i = 1, 2, \cdots, n$, to the data points $(t_j, [y]_j)$ $j = 1, 2, \cdots, m$ and $t_j \in \mathbb{R}^n$, $[y]_j \in \mathbb{R}$. The function $f(t)$ is to fit the data in the least-squares sense. A further restriction is that each residual $[r]_j = f(t_j) - [y]_j$ must satisfy $|[r]_j| \leq \epsilon$. It is clear that these additional requirements lead to an inconsistent set of $2m$ linear inequalities in the parameters $x = (x_1, x_2, \cdots, x_n)^T$. It is debatable whether this problem deserves an approximate solution. We will not argue that point here. But given that we want to resolve these conflicting inequalities, we introduce slack variables w_j into the inequalities. This problem may be written as a constrained linear least-squares problem as follows:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{j=1}^{m} (f(t_j) - [y]_j)^2 \\ \text{subject to} \quad & |f(t_j) - [y]_j| \leq \epsilon, \quad \text{for j} = 1, 2, \cdots, m, \end{aligned} \tag{5.1}$$

where the parameters $0 \leq [t_j]_i \leq 1$, and $[y]_j$ for $j = 1, \cdots m$ are given. This optimization problem is equivalent to

$$\begin{aligned} \underset{x,w}{\text{minimize}} \quad & \tfrac{1}{2}\|Ax - y\|^2 \\ \text{subject to} \quad & Ax + \delta w = \epsilon - y, \quad Ax - \delta w = -\epsilon + y, \end{aligned}$$

where $A = \begin{bmatrix} [t_1]_1 & [t_2]_1 & \cdots & [t_m]_1 \\ \vdots & \vdots & \cdots & \vdots \\ [t_1]_n & [t_2]_n & \cdots & [t_m]_n \end{bmatrix}$, $x = (x_1, x_2, \cdots, x_n)$, $w = (w_1, w_2, \cdots, w_m)$, and $y = ([y]_1, \cdots, [y]_m)$.

5.2 Large-Scale ℓ-Regularized LS Problems

A lot of attention has been paid to ℓ-regularization based methods for sparse signal reconstruction, e.g., basis pursuit denoising and compressed sensing. These problems can be cast as ℓ-regularized LS problems, which can be reformulated as convex quadratic programs, and then solved by several standard methods such as interior-point methods. For small and medium sized problems, the primal-dual interior-point method can be used. In general, the interior-point method can solve large sparse problems, with a million variables and observations, in a few minutes.

Here, we introduce this kind of problem as an application to our approach, although we will not go into the details. For further details, see e.g. Kim et al. (2007a). Suppose that we have a linear model of the form

$$d = Ax + v,$$

where $x \in \mathbb{R}^p$ is the vector of unknowns, $d \in \mathbb{R}^p$ is the vector of observations, $v \in \mathbb{R}^p$ is the noise, and $A \in \mathbb{R}^{p \times n}$ is the data matrix. When $p \geq n$ we can determine x by solving the least-squares problem of minimizing the quadratic loss $\frac{1}{2}\|Ax - d\|^2$. A standard technique to prevent over-fitting is ℓ regularization, which can be written as

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - d\|^2 + \delta\|x\|_\ell, \tag{5.2}$$

where $\delta > 0$ is the regularization parameter. Another technique is to use ℓ_2-regularization, which is called Tikhonov regularization, and can be written as

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - d\|^2 + \delta\|x\|_2^2, \tag{5.3}$$

where $\delta > 0$ is the regularization parameter. The ℓ_2-regularized least-squares program has the analytic solution

$$x^* = (A^T A + \delta I)^{-1} A^T d.$$

The solution to the ℓ_2- regularization problem can be computed by direct methods, which require $\mathcal{O}(n^3)$ flops. The solution can also be computed by applying iterative (non-direct)

methods (e.g., the conjugate gradient method) to the linear system of equations

$$(AA^T + \delta I)x = A^T d.$$

Iterative methods are efficient, especially when there are fast algorithms for the matrix-vector multiplications with the data matrix A and its transpose A^T (i.e., Au and $A^T v$ with $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^m$), which is the case when A is sparse or has a special form such as partial Fourier and wavelet matrices.

5.3 LS with ℓ_1-Norm Regularization

As another application, we consider a LS problem with ℓ_1-norm regularization. In ℓ_1-regularized LS Problem, we substitute a sum of absolute values for the sum of squares used in Tikhonov regularization, to obtain

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - d\|^2 + \delta\|x\|_1, \tag{5.4}$$

where $\|x\|_1 = \sum_{i=1}^{n} |[x]_i|$ denotes the ℓ_1 norm of x and $\delta > 0$ is the regularization parameter. This kind of problem always has a solution, but it need not be unique. Some problems do not have the standard form (5.4) but have a more general form

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - d\|^2 + \sum_{i=1}^{n} [\delta]_i |[x]_i|, \tag{5.5}$$

where $[\delta]_i \geq 0$ are regularization parameters. The variables $[x]_i$ that correspond to $[\delta]_i = 0$ are not regularized. The ℓ_1-norm LS problem can be transformed to a convex quadratic problem with linear equality constraints. The equivalent LS problem can be solved by standard convex optimization methods such as interior-point methods. It can readily handle small and medium-sized problems. Standard methods cannot efficiently handle large-scale problems in which there are fast algorithms for the matrix-vector operations with A and A^T. Specialized interior-point methods that exploit such algorithms can scale to large problems, as demonstrated by Chen et al. (2001) and Johnson et al. (2000). The method of chapter 3 also can handle large-scale problems. Now, we show how we can transform (5.5) to a convex quadratic problem.

Theorem 5.3.1. *Let f and g_i for $i = 1, 2, \cdots, m$, be real-valued with $dom(f)$, $dom(g_i)$ and $C \subseteq \mathbb{R}^n$. The two following programs are equivalent*

$$\underset{x}{\textit{minimize}} \quad f(x) + g_1(x) + \cdots + g_m(x) \quad \textit{subject to} \quad x \in C \tag{5.6a}$$

$$\begin{aligned} &\underset{t,u_1,\cdots,u_m,x}{\textit{minimize}} \quad t + u_1 + \cdots + u_m \\ &\textit{subject to} \quad f(x) \le t, \quad g_i(x) \le u_i, \ \textit{for } i = 1, 2, \cdots, m, \quad x \in C \end{aligned} \tag{5.6b}$$

Proof. The second problem (5.6b) looks like the epigraph of (5.6a). Problem (5.6a) is a problem in $\mathbb{R}^n$ and (5.6b) is in $\mathbb{R}^{n+m+1}$. We show that $x^* \in \mathbb{R}^n$ solves (5.6a) if and only if $(t^*, u_1^*, \cdots, u_m^*, x^*) \in \mathbb{R}^{n+m+1}$ solves (5.6b), where $t^* = f(x^*)$ and $u_i^* = g_i(x^*)$ for i = $1, 2, \cdots, m$. Let x^* solve (5.6a) and $t^* = f(x^*)$, $u_i^* = g_i(x^*)$ for i = $1, 2, \cdots, m$. We claim that $(t^*, u_1^*, \cdots, u_m^*, x^*)$ is an optimal solution of (5.6b). For all $(t, u_1, \cdots, u_m, x)$ feasible for (5.6b) we have

$$f(x) + g_1(x) + \cdots + g_m(x) \le t + u_1 + \cdots + u_m \quad \forall x \in C.$$

Therefore,

$$f(x^*) + g_1(x^*) + \cdots + g_m(x^*) \le f(x) + g_1(x) + \cdots + g_m(x) \le t + u_1 + \cdots + u_m \quad \forall x \in C,$$

i.e.,

$$t^* + u_1^* + \cdots + u_m^* \ \le t + u_1 + \cdots + u_m.$$

Now, let $w^* = (t^*, u_1^*, \cdots, u_m^*, x^*)$ solve (5.6b). By feasibility of w^* we have

$$f(x^*) \le t^*,$$

$$g_i(x^*) \le u_i^*, \qquad \text{for i} = 1, 2, \cdots, m.$$

Since $(f(x), g_1(x), \cdots, g_m(x), x)$ is feasible for (5.6b), we get

$$f(x^*) + g_1(x^*) + \cdots + g_m(x^*) \le t^* + u_1^* + \cdots + u_m^* \le f(x) + g_1(x) + \cdots + g_m(x) \quad \forall x \in C.$$

This completes the proof of the theorem. □

A special case is when $g(x) = \|x\|_1 = \sum_{i=1}^{n} |[x]_i|$, which leads to

$$\underset{x}{\text{minimize}} \quad f(x) + \sum_{i=1}^{n} r_i \quad \text{subject to} \quad |[x]_i| \leq r_i \quad \forall i = 1, 2, \cdots, n. \tag{5.7}$$

Now, (5.7) is equivalent to

$$\underset{x}{\text{minimize}} \quad f(x) + e^T r \quad \text{subject to} \quad -r \leq x \leq r. \tag{5.8}$$

In another words we can say that by moving the ℓ_1 norm in (5.3) to the constraints we have

$$\underset{x,r}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - d\|^2 + \delta e^T r \quad \text{subject to} \quad -r \leq x \leq r. \tag{5.9}$$

Using theorem 5.3.1, it is easy to show that (5.9) is equivalent to (5.4). The former program becomes

$$\underset{x,r,t}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - d\|^2 + \delta e^T r \quad \text{subject to} \quad \begin{pmatrix} -I & -I & I \\ I & -I & I \end{pmatrix} \begin{pmatrix} x \\ r \\ t \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad t \geq 0.$$

We give a simple example of (5.4), then apply the methods of Chapter 2 and 3 on large-scale sparse signal reconstruction problems. We tested our methods versus $\ell_1 - \ell_s$ (Kim et al. (2007a)), CPLEX (CPLEX (2007)), IPOPT (Waechter (2007)), and LOQO (Vanderbei (1999)) with a special kind of matrix called 2-D discrete cosine transform (DCT2) that is used in image compression. DCTs are important in numerous applications and in data compression. DCT is especially used in audio (e.g. MP3) and in colour or grey scale images (e.g. JPEG).

We use a standard MATLAB routine called `dct2.m` to generate DCT2 matrices. Suppose matrix A is given by DCT2, set x_0 as a random vector and $d = Ax_0$. Now, we try to solve the following optimization problem to find a sparser optimal solution with an optimal value equal to zero

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - d\|^2 + \delta \sum_{i=1}^{n} |[x]_i|. \tag{5.10}$$

Given

$$A = \begin{pmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & 0.2 & 0.3 \\ 0 & 0.1 & 1 & 0.2 \\ 1 & 0 & 1 & 1 \end{pmatrix},$$

we set $x_0 = (1, 0, 1, 0)$, $d = Ax_0$, and $\delta = 0.01$. Using $\ell_1 - \ell_s$ (Kim et al. (2007a)), we get the optimal objective value $1.990410e - 02$, solve time 0.160 seconds, and the minimizer:

$$x = (0.9877, 0.0007, 0.9891, 0.0046).$$

For this point on, we refer to the method of Chapter 2 as 3×3 method and to the method of Chapter 3 as the 4×4. The 4×4 method, we obtain the optimal objective value $1.990227e-02$, solve time 0.012 seconds, and the minimizer:

$$x = (0.9900, 0.0000, 0.9904, 0.0000).$$

Using the 3×3 method we obtain the optimal objective value $1.990244e - 02$, solve time 0.007 seconds, and the minimizer:

$$x = (0.9899, 0.0000, 0.9903, 0.0001).$$

Table 5.1 reports results of applying (Un)Scaled3×3, (Un)Scaled4×4, and $\ell_1 - \ell_s$ implementations . The columns are, from left to right, the problem name, number of iterations, final objective value, KKT residual, solve time, and number of nonzero elements in the minimizer. We can see that the (Un)Scaled4×4 and (Un)Scaled3×3 achieve a sparser and better solution in this simple example. For this example there is no significant difference between the scaled and unscaled forms. The scaled method automatically scaled the problem prior to solution so as to equilibrate the rows and columns of the constraints. Equilibration is done by first dividing every row by its largest element in absolute value and then by dividing every column by its largest element in absolute value. An advantage of this method is that it is not sensitive to dense columns. This property makes the method typically more robust than a standard implementation and the linear system solves often much faster than in a traditional interior-point method in augmented form. Scaling is especially effective on large-scaled problems, as shown in the next section. $\ell_1 - \ell_s$ is an inexact Newton interior-point method specifically designed for (5.9) also described by Kim et al. (2007a). In $\ell_1 - \ell_s$, a primal barrier method is applied to (5.4) and approximate Newton steps are computed using the preconditioned conjugate-gradient method. In particular, the Hessian contains a term of the form $A^T A$ and this term is preconditioned by its diagonal.

5.4 Application to Sparse Signal Recovery

We can denote any discrete signal or data as a real vector x in a high dimensional space. In general, sparse vectors are vectors for which most components are zero. To quantitatively

Table 5.1 A simple example of $\ell_1 - \ell_s$

Name	Iter	Cost	KKTresidual	Time	Nnz–Minimizer
UnScaled-4x4	10	1.990227e-02	6.887167e-09	1.49e-02	50%
Scaled-4x4	10	1.990227e-02	6.887167e-09	1.17e-02	50%
UnScaled-3x3	7	1.990244e-02	8.164790e-09	8.02e-03	75%
Scaled-3x3	7	1.990244e-02	8.164790e-09	7.70e-03	75%
l1-ls	16	1.990411e-02	1.439952e-05	5.70e-01	100%

describe this, for an integer k we call a vector k-sparse if at most k of its components are non-zero. Formally, a vector $x \in \mathbb{R}^n$ is k-sparse if $\|x\|_0 = k$ where $\|x\|_0 = \sum_{i=1}^n [x]_i^0$ (with the convention $0^0 = 0$). We consider sparse signal recovery problems with matrices A of size 4096×1024 and vectors $x \in \mathbb{R}^{1024}$ with sparsity $\|x\|_0 = k$ where $k = 100$ and $k = 150$. For the given matrix A the vector $Ax = d$ is known. The measurement matrix A is generated by the command `A = rand(4096, 1024)`. Let x be a Gaussian random vector whose entries are independent identically distributed (i.i.d.). We solve (5.4) with the 3×3 and 4×4 methods and compare the result, the successful sparse recovery rate, with that of the $\ell_1 - \ell_s$ method. Here, we also set the parameter $\delta = 0.01$. From these results, we can see that our method can perform better than the $\ell_1 - \ell_s$ method when we expect sparsity in the optimal solution.

5.5 Generation of Test Problems

We generate data for the following least-squares problem

$$\underset{x}{\text{minimize}} \quad \tfrac{1}{2}\|Ax - d\|^2 \quad \text{subject to} \quad Bx \geq b, \tag{5.11}$$

where:

$$\begin{aligned}
&A = (q_{ij}), \quad q_{ij} \in (-10, 10) \text{ randomly chosen }, i = 1, \cdots, p, \quad j = 1, \cdots, n, \\
&d = (d_i), \quad d_i = \sum_{j=1}^{n} q_{ij}, \text{ for } i = 1, \cdots, p, \\
&B = (q_{ij}), \quad q_{ij} \in (-3, 3) \text{ randomly chosen}, i = 1, \cdots, m, \quad j = 1, \cdots, n, \\
&b_i = \begin{cases} \sum_{j=1}^{n} q_{ij} & \text{if } i = 1, \\ \sum_{j=1}^{n} q_{ij} - m\mu_i & \text{if } i \neq 1, \end{cases}
\end{aligned} \tag{5.12}$$

with $\mu_i \in (0, 1)$, randomly chosen. The exact solution x^* of these problems is given by

$$x^* = (1, \cdots, 1).$$

The first constraint is active at x^* and the exact minimal value of the least-squares problem is zero. The subsequent examples are chosen such that the exact solution x^* of LS problem is known. So it is possible to compare the number of exact digits in the comparison with other solvers.

5.6 Numerical Comparison of Four Algorithms Solving LS

We now compare (Un)Scaled-4×4 and (Un)scaled-3×3 with three different algorithms: $\ell_1 - \ell_s$, IPOPT, and LOQO for the numerical solution of (2.11). Tables 5.2–5.17 show the results of this comparison. The columns are, from left to right, the problem name, number of iterations, final objective value, KKT residual, and solve time. We generate a DCT2 matrix and solve (5.10). All stopping tolerances are set to 10^{-6} and presolve is turned off. In Figures 5.6–5.6, we illustrate the performance of our approach against four comparable and closely related solvers using performance profiles (Dolan and Moré, 2002). Problem names are **p-n-m-solver name**, where $p \times n$ and $n \times m$ denote the dimensions of the least-squares and constraint matrices, respectively. If $n = m$, problem names are **p-n-solver name**.

A brief overview of optimization solvers that we use for comparison

Ipopt (Interior Point OPTimizer) is a software package for large-scale nonlinear optimization. It is designed to find (local) solutions. The objective function can be nonlinear and nonconvex, but should be twice continuously differentiable. Ipopt is written in C++ and is released as open source code.

LOQO is a system for solving smooth nonlinear constrained optimization problems. LOQO is based on an infeasible, primal-dual, interior-point method applied to a sequence of QP.

Cplex is one of the most powerful commercial solvers, capable of using several CPU in parallel calculations. Problem types that Cplex can handle: LP, MILP, (MI)QP, (MI)QCQP, (MI)SOCP (and some more).

$\boldsymbol{\ell_1 - \ell_s}$ is a Matlab implementation of the interior-point method for regularized least-squares described in Kim et al. (2007a). $\ell_1 - \ell_s$ is developed for large problems. It can solve large sparse problems. It can also efficiently solve very large dense problems, that arise in sparse signal recovery with orthogonal transforms, by exploiting fast algorithms for these transforms.

Table 5.2 Comparison for problem with $p = 8$ and $n = 12$

Name	Iter	Cost	KKTresidual	Time
8-12–UnScaled-4x4	5	4.415775e-01	4.758201e-08	0.006
8-12–Scaled-4x4	5	4.436060e-01	3.790156e-07	0.006
8-12–UnScaled-3x3	5	4.435796e-01	1.333960e-07	0.006
8-12–Scaled-3x3	5	4.435796e-01	2.379830e-07	0.006
8-12–l1-ls	16	4.436101e-01	4.200664e-04	0.397
8-12–IPOPT	7	4.435786e-01	2.506328e-14	0.012
8-12–LOQO	14	4.435786e-01	1.700000e-10	0.002

Table 5.3 Comparison for problem with $p = 32$ and $n = 8$

Name	Iter	Cost	KKTresidual	Time
32-8–UnScaled-4x4	8	5.039810e-02	4.526520e-12	0.012
32-8–Scaled-4x4	6	5.064889e-02	1.087014e-08	0.008
32-8–UnScaled-3x3	6	5.064889e-02	1.192544e-08	0.008
32-8–Scaled-3x3	6	5.064889e-02	2.914584e-08	0.008
32-8–l1-ls	18	5.064888e-02	4.426570e-05	0.205
32-8–IPOPT	5	5.064889e-02	2.505982e-14	0.010
32-8–LOQO	11	5.064889e-02	1.500000e-09	0.003

Table 5.4 Comparison for problem with $p = 16$ and $n = 24$

Name	Iter	Cost	KKTresidual	Time
16-24–UnScaled-4x4	8	1.016159e-01	1.132528e-10	0.012
16-24–Scaled-4x4	7	1.018095e-01	2.018836e-08	0.010
16-24–UnScaled-3x3	6	1.018206e-01	1.304466e-08	0.008
16-24–Scaled-3x3	6	1.018206e-01	4.551537e-08	0.008
16-24–l1-ls	16	1.018058e-01	9.485795e-05	0.250
16-24–IPOPT	6	1.018034e-01	2.505982e-14	0.012
16-24–LOQO	12	1.018034e-01	1.500000e-09	0.003

Table 5.5 Comparison for problem with $p = 64$ and $n = 16$

Name	Iter	Cost	KKTresidual	Time
64-16–UnScaled-4x4	7	8.333846e-02	9.663905e-09	0.012
64-16–Scaled-4x4	7	8.343721e-02	3.657231e-10	0.011
64-16–UnScaled-3x3	6	8.343960e-02	2.416399e-08	0.009
64-16–Scaled-3x3	6	8.343954e-02	1.213753e-07	0.010
64-16–l1-ls	19	8.343721e-02	5.338899e-05	0.337
64-16–IPOPT	6	8.343723e-02	2.505982e-14	0.014
64-16–LOQO	11	8.343722e-02	4.000000e-09	0.005

Table 5.6 Comparison for problem with $p = 128$ and $n = 32$

Name	Iter	Cost	KKTresidual	Time
128-32–UnScaled-4x4	11	1.022355e-01	2.304974e-10	0.028
128-32–Scaled-4x4	6	1.023510e-01	6.286439e-07	0.014
128-32–UnScaled-3x3	6	1.032129e-01	6.890546e-08	0.015
128-32–Scaled-3x3	6	1.032304e-01	8.811947e-07	0.015
128-32–l1-ls	17	1.022411e-01	9.131019e-05	0.195
128-32–IPOPT	6	1.022411e-01	2.505982e-14	0.026
128-32–LOQO	12	1.022411e-01	8.200000e-10	0.022

Table 5.7 Comparison for problem with $p = 256$ and $n = 64$

Name	Iter	Cost	KKTresidual	Time
256-64–UnScaled-4x4	9	1.547477e-01	6.786164e-12	0.059
256-64–Scaled-4x4	6	1.550747e-01	9.862409e-07	0.030
256-64–UnScaled-3x3	6	1.551046e-01	6.123953e-09	0.033
256-64–Scaled-3x3	6	1.551061e-01	1.731912e-07	0.034
256-64–l1-ls	20	1.547542e-01	1.219684e-04	0.340
256-64–IPOPT	8	1.547542e-01	1.776357e-14	0.087
256-64–LOQO	13	1.547542e-01	2.300000e-10	0.134

Table 5.8 Comparison for problem with $p = 1024$ and $n = 256$

Name	Iter	Cost	KKTresidual	Time
1024-256–UnScaled-4x4	12	6.069454e-01	9.750109e-10	0.849
1024-256–Scaled-4x4	6	6.086937e-01	3.062681e-07	0.401
1024-256–UnScaled-3x3	4	9.228976e-01	8.034770e-07	0.285
1024-256–Scaled-3x3	7	6.081951e-01	1.112049e-07	0.496
1024-256–l1-ls	20	6.069323e-01	4.734477e-04	0.612
1024-256–IPOPT	10	6.069326e-01	2.842171e-14	2.873
1024-256–LOQO	15	6.069323e-01	8.500000e-10	5.444

Table 5.9 Comparison for problem with $p = 1072$ and $n = 768$

Name	Iter	Cost	KKTresidual	Time
1072-768–UnScaled-4x4	11	1.912995e+00	5.778455e-12	2.955
1072-768–Scaled-4x4	6	1.921405e+00	6.686542e-07	1.614
1072-768–UnScaled-3x3	3	5.039887e+00	9.130975e-07	0.874
1072-768–Scaled-3x3	6	2.107093e+00	6.404760e-07	1.595
1072-768–l1-ls	21	1.913003e+00	1.551190e-03	1.243
1072-768–IPOPT	10	1.913004e+00	1.136868e-13	9.275
1072-768–LOQO	16	1.913003e+00	1.300000e-10	42.021

Table 5.10 Comparison for problem with $p = 1576$ and $n = 634$

Name	Iter	Cost	KKTresidual	Time
1576-634–UnScaled-4x4	11	1.598491e+00	5.310856e-07	3.007
1576-634–Scaled-4x4	6	1.605521e+00	1.999222e-07	1.658
1576-634–UnScaled-3x3	3	4.236755e+00	6.896229e-07	0.827
1576-634–Scaled-3x3	6	1.839050e+00	8.566569e-07	1.653
1576-634–l1-ls	23	1.598489e+00	1.285429e-03	1.221
1576-634–IPOPT	10	1.598490e+00	2.842171e-14	16.026
1576-634–LOQO	15	1.598489e+00	1.800000e-09	45.298

Table 5.11 Comparison for problem with $p = 2048$ and $n = 512$

Name	Iter	Cost	KKTresidual	Time
2048-512–UnScaled-4x4	12	1.266813e+00	5.560368e-09	3.510
2048-512–Scaled-4x4	6	1.272650e+00	1.765203e-07	2.111
2048-513–UnScaled-3x3	3	3.379576e+00	5.885186e-07	0.907
2048-512–Scaled-3x3	6	1.441661e+00	7.486843e-07	1.770
2048-512–l1-ls	21	1.266812e+00	8.485015e-04	1.252
2048-512–IPOPT	1	1.266812e+00	1.000000e+00	0.572
2048-512–LOQO	15	1.266812e+00	1.800000e-10	42.958

Table 5.12 Comparison for problem with $p = 2144$ and $n = 1536$

Name	Iter	Cost	KKTresidual	Time
2144-1536–UnScaled-4x4	12	3.823373e+00	3.980785e-10	22.817
2144-1536–Scaled-4x4	6	3.840569e+00	1.733043e-07	10.909
2144-1536–UnScaled-3x3	3	1.029325e+01	3.267540e-07	5.369
2144-1536–Scaled-3x3	7	3.831230e+00	1.768381e-08	13.402
2144-1536–l1-ls	25	3.823393e+00	3.079818e-03	3.569
2144-1536–IPOPT	10	3.823395e+00	3.410605e-13	68.007
2144-1536–LOQO	15	3.823393e+00	2.100000e-09	334.691

Table 5.13 Comparison for problem with $p = 1536$ and $n = 1072$

Name	Iter	Cost	KKTresidual	Time
2288-1072–UnScaled-4x4	11	2.772483e+00	1.099068e-10	11.028
2288-1072–Scaled-4x4	6	2.782258e+00	1.740290e-07	6.150
2288-1072–UnScaled-3x3	3	7.241645e+00	3.645765e-07	2.774
2288-1072–Scaled-3x3	7	2.777928e+00	1.684233e-08	6.472
2288-1072–l1-ls	21	2.772473e+00	2.166878e-03	2.225
2288-1072–IPOPT	10	2.772474e+00	1.705303e-12	55.683
2288-1072–LOQO	16	2.772473e+00	1.300000e-10	193.910

Table 5.14 Comparison for problem with $p = 3192$ and $n = 2048$

Name	Iter	Cost	KKTresidual	Time
3192-2048–UnScaled-4x4	12	5.047011e+00	1.361515e-12	52.103
3192-2048–Scaled-4x4	6	5.079541e+00	1.516105e-07	25.994
3192-2048–UnScaled-3x3	3	1.387182e+01	1.913808e-07	13.731
3192-2048–Scaled-3x3	7	5.056596e+00	1.425411e-08	30.298
3192-2048–l1-ls	23	5.047030e+00	3.383664e-03	5.695
3192-2048–IPOPT	1	5.047030e+00	1.000000e+00	9.803
3192-2048–LOQO	16	5.047030e+00	5.700000e-10	936.265

Table 5.15 Comparison for problem with $p = 1536$ and $n = 2192$

Name	Iter	Cost	KKTresidual	Time
3768-2192–UnScaled-4x4	10	5.499013e+00	5.086452e-07	54.753
3768-2192–Scaled-4x4	6	5.521418e+00	1.362809e-07	32.843
3768-2192–UnScaled-3x3	3	1.481956e+01	1.560411e-07	19.010
3768-2192–Scaled-3x3	7	5.509311e+00	1.151996e-08	43.067
3768-2192–l1-ls	23	5.498724e+00	3.631953e-03	7.614
3768-2192–IPOPT	1	5.498724e+00	1.000000e+00	12.851
3768-2192–LOQO	16	5.498724e+00	6.400000e-10	1266.820

Table 5.16 Comparison for problem with $p = 4096$ and $n = 1024$

Name	Iter	Cost	KKTresidual	Time
4096-1024–UnScaled-4x4	11	2.507106e+00	6.057688e-13	17.565
4096-1024–Scaled-4x4	6	2.526827e+00	1.415525e-07	9.638
4096-1024–UnScaled-3x3	3	6.884260e+00	2.106426e-07	4.986
4096-1024–Scaled-3x3	7	2.512113e+00	9.796640e-09	11.317
4096-1024–l1-ls	22	2.507109e+00	2.048682e-03	3.107
4096-1024–IPOPT	1	2.507109e+00	1.000000e+00	6.477
4096-1024–LOQO	16	2.507109e+00	8.000000e-10	359.174

Table 5.17 Comparison for problem with $p = 4152$ and $n = 2288$

Name	Iter	Cost	KKTresidual	Time
4152-2288–UnScaled-4x4	12	5.768351e+00	7.942647e-12	82.323
4152-2288–Scaled-4x4	6	5.791925e+00	1.340150e-07	41.158
4152-2288–UnScaled-3x3	3	1.548851e+01	1.383680e-07	22.056
4152-2288–Scaled-3x3	7	5.779492e+00	1.059316e-08	50.750
4152-2288–l1-ls	23	5.768362e+00	4.532695e-03	8.096
4152-2288–IPOPT	1	5.768362e+00	1.000000e+00	11.931
4152-2288–LOQO	16	5.768362e+00	4.500000e-10	1565.130

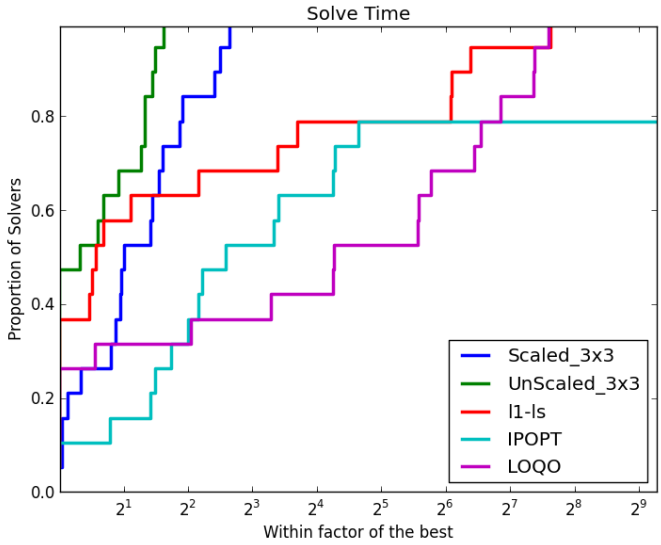

Figure 5.1 Performance in Terms of Time Using 3x3

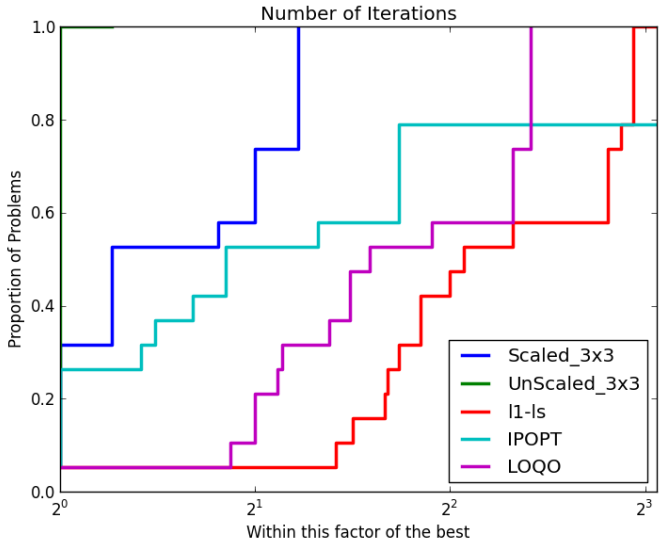

Figure 5.2 Performance in Terms of Number of Iterations Using 3x3

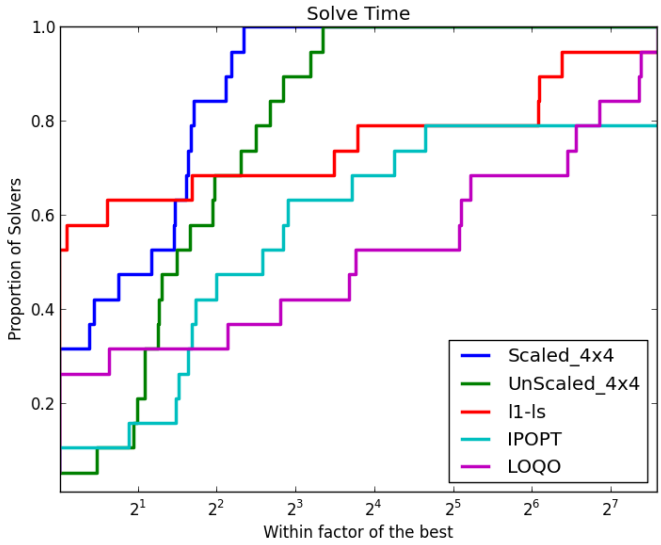

Figure 5.3 Performance in Terms of Time Using 4x4

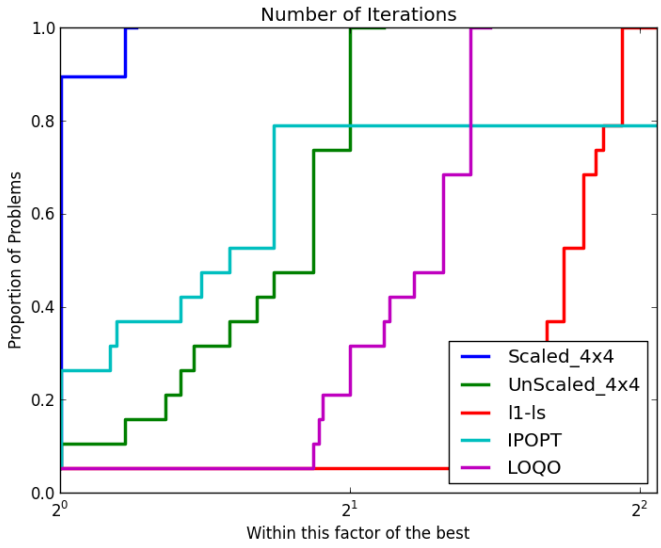

Figure 5.4 Performance in Terms of Number of Iterations Using 4x4

If we generate the data defined by (5.12), the table 5.18 shows the results of the 3×3 method and that of LOQO, CPLEX and IPOPT.

Table 5.18 Randomly generated data defined by (5.12)

Problems	3X3 Solver			LOQO Solver			CPLEX Solver			IPOPT Solver		
p_n	Iter	Cost	Time	Iter	Cost	Time	Iter	Cost	Time	Iter	Cost	Time
5_17	04	2.279928e-16	0.004	08	1.596327e-28	0.002	04	1.577427e-17	0.000	03	1.247657e-22	0.009
10_34	01	6.021708e-13	0.001	08	8.686748e-29	0.006	04	7.678447e-18	0.001	03	1.782053e-22	0.009
15_51	04	4.473841e-17	0.006	09	6.845116e-26	0.017	06	1.624245e-25	0.002	05	6.688473e-25	0.015
20_68	01	1.779093e-13	0.002	09	2.495789e-25	0.064	06	1.905625e-25	0.002	05	4.502411e-25	0.019
25_85	01	6.326390e-13	0.002	09	2.380146e-25	0.033	06	2.782475e-25	0.003	05	1.529516e-25	0.026
30_102	01	7.767067e-14	0.004	09	1.060389e-24	0.098	06	6.264471e-22	0.004	05	2.015141e-25	0.034
35_119	01	3.088590e-13	0.003	09	6.565800e-26	0.153	06	1.699474e-25	0.005	05	7.511074e-26	0.047
40_136	01	6.786264e-13	0.004	09	6.085394e-25	0.232	06	2.691303e-25	0.007	05	8.488489e-26	0.063
45_153	01	3.163095e-13	0.006	10	9.660471e-25	0.377	06	1.428061e-22	0.008	05	5.645330e-26	0.082
50_170	01	1.750154e-13	0.006	09	1.596443e-23	0.465	06	1.074870e-22	0.010	05	4.561103e-26	0.100
55_187	01	1.287814e-13	0.008	10	1.981103e-25	0.688	06	6.332380e-23	0.013	05	8.044326e-27	0.129
60_204	01	2.040475e-14	0.011	10	1.642710e-26	0.946	07	1.070610e-26	0.016	05	2.734176e-26	0.162
65_221	01	4.444716e-14	0.013	10	1.653470e-25	1.201	06	2.589970e-23	0.017	05	1.615499e-26	0.193
70_238	01	6.508670e-14	0.015	10	1.086136e-25	1.602	07	4.873302e-27	0.028	05	2.101545e-26	0.237
75_255	01	1.319467e-13	0.017	11	8.733719e-26	2.124	06	4.433635e-23	0.025	05	6.376042e-27	0.278
80_272	01	6.018428e-13	0.018	10	1.609376e-24	2.058	06	6.799117e-23	0.028	05	1.349015e-26	0.341
85_289	01	2.040109e-13	0.024	10	5.470410e-25	2.714	06	3.318592e-23	0.036	05	1.372251e-26	0.396
90_306	01	1.888020e-13	0.025	11	3.339994e-26	3.323	06	2.201465e-23	0.039	05	6.976566e-27	0.449
95_323	01	3.303506e-14	0.030	11	5.430104e-26	3.440	06	1.158303e-23	0.044	06	2.530921e-27	0.602
100_340	01	1.274011e-13	0.032	10	9.933018e-25	5.180	06	1.999058e-23	0.051	05	5.881568e-27	0.643
105_357	01	7.088517e-14	0.037	10	1.313937e-24	5.089	06	1.433046e-23	0.056	05	5.742792e-27	0.688
110_374	01	1.307275e-13	0.041	10	2.516873e-24	6.049	06	2.847833e-23	0.065	05	5.180019e-27	0.773
115_391	01	7.559279e-14	0.050	11	5.255312e-25	6.176	06	1.700838e-23	0.070	06	2.313449e-27	0.994
120_408	01	5.914629e-14	0.050	10	2.594363e-24	7.240	06	1.729012e-23	0.084	05	2.789481e-27	0.937
125_425	01	3.102802e-14	0.055	10	8.713273e-25	7.754	07	8.588059e-28	0.105	05	5.813813e-27	1.050
130_442	01	1.831899e-13	0.070	11	3.340932e-25	9.207	06	7.406034e-24	0.101	06	1.936553e-27	1.402
135_459	01	3.246015e-14	0.077	11	1.221210e-25	10.661	06	8.006077e-24	0.117	06	2.086028e-27	1.907
140_476	01	9.007068e-14	0.084	11	2.828252e-25	12.111	06	8.065670e-24	0.118	06	2.128748e-27	1.604
145_493	01	1.325716e-13	0.085	12	3.041239e-26	14.381	06	8.783337e-24	0.129	06	1.683732e-27	1.837

CHAPTER 6

CONCLUSION

6.1 Summary of Work

In this work, we consider constrained linear least-squares problems in standard form

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ c^Tx + \tfrac{1}{2}\|Ax-d\|^2 \quad \text{subject to}\ Bx = b,\ x \geq 0. \tag{6.1}$$

An interior-point method applied directly to (6.1) might encounter several difficulties. We remove the difficulties in different ways and obtain two slightly different implementations. We consider the following regularization of (6.1) proposed by Friedlander and Orban (2012):

$$\begin{aligned} \underset{x\in\mathbb{R}^n, w\in\mathbb{R}^m}{\text{minimize}} \quad & c^Tx + \tfrac{1}{2}\|Ax-d\|^2 + \tfrac{1}{2}\rho\|x-x_k\|^2 + \tfrac{1}{2}\delta\|w+y_k\|^2 \\ \text{subject to} \quad & Bx + \delta w = b,\ x \geq 0. \end{aligned} \tag{6.2}$$

At each iteration, a step is computed by solving a large and sparse symmetric quasi-definite linear system (Vanderbei, 1995). Contrary to most interior-point implementations, partial block elimination is not applied to this system to reduce it to the so-called *augmented system* form or to the *normal equations*. Instead, a similarity transformation that guarantees that the system remains uniformly bounded and nonsingular throughout the iterations is applied. We establish global convergence under weak assumptions. In particular, no assumption on the rank of B or A is made. A distinctive feature of the regularization (6.2) is that it enables a solution of (6.1) to be recovered in many situations, and not only a solution to a perturbed problem. In addition, (6.2) is never solved to optimality for fixed values of ρ, δ, x_k and y_k. Instead, it is used to compute a single Newton step before attention shifts to the next regularized subproblem.

Our method is implemented in the Python language as part of the NLPy library (Orban, 2012). We illustrate the performance of our approach against $\ell_1-\ell_s$ on sparse single recovery problems generated randomly following the procedure given by Kim et al. (2007b). We try to find a point that is guaranteed to be no more than one percent suboptimal, i.e., the regularization parameter was taken as $\delta = 0.01$. All large-scale problems, scaled form achieves significantly better performance in terms of number of iterations, run time, and sparsity of minimizer.

6.2 Limitations of the Proposed Solution and Future Improvements

Many applications only provide A and B in the form of linear operators instead of explicit matrices in (6.1) or they have very large dimensions. Using sparse LDL^T factorization in the case of large dimensions requires a lot of memory. Iterative methods specialized to symmetric quasi-definite systems have been recently proposed by Arioli and Orban (2012). Our algorithm paves the way to a matrix-free implementation using such iterative methods. This yields an elegant framework in which an unconstrained regularized linear least-squares problem must be solved at each iteration. Other future improvements include the solution of *constrained nonlinear least-squares* problems.

6.2.1 Nonlinear Least-Squares with Linear Constraints

In this section, we consider a regularization technique for nonlinear least-squares problems with linear constraints. Nonlinear least-square problems can be written as:

$$\min_{x \in \mathbb{R}^n} \ \tfrac{1}{2} \sum_{i=1}^{m} f_i(x)^2 \quad \text{subject to } Ax = b, \ x \geq 0, \tag{6.3}$$

where each function $f_i : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. Numerical difficulties can arise when the matrix A and/or the Jacobian of $F : \mathbb{R}^n \to \mathbb{R}^m$, $F(x) := (f_1(x), \ldots, f_m(x))$ do not have full row rank. We propose a primal-dual interior point method. New challenges may occur if at a solution x^*, some components are such that $x_i^* = 0$ and $z_i^* = 0$, where z^* is the vector of dual variables associated with the non-negativity constraints $x \geq 0$. We separate our methodology into numerical and theoretical considerations.

6.2.2 Numerical Aspects

The application of an interior-point method to (6.3) gives us the perturbed optimality conditions

$$\begin{bmatrix} J(x)^T F(x) - A^T y - z \\ A(x) - b \\ Xz - \mu e \end{bmatrix} = 0, \qquad (x, z) > 0, \tag{6.4}$$

where $\mu > 0$ is the barrier parameter, $J(x)$ is the Jacobian of $F(x)$, y is the vector of Lagrange multipliers associated with the equality constraints, and $X = \operatorname{diag}(x)$. The calculation of the

Newton step $(\Delta x, \Delta y, \Delta z)$ for (6.4) requires solving the linear system

$$\begin{bmatrix} H(x) & A^T & -I \\ A & 0 & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ -\Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} J(x)^T F(x) - A^T y - z \\ A(x) - b \\ Xz - \mu e \end{bmatrix}, \tag{6.5}$$

where $H(x) = J(x)^T J(x) + \sum_{i=1}^m f_i(x) \nabla^2 f_i(x)$ is the Hessian of the Lagrangian of (6.3). After elimination of Δz, we need to solve the following symmetric system

$$\begin{bmatrix} H(x) + X^{-1}Z & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ -\Delta y \end{bmatrix} = - \begin{bmatrix} J(x)^T F(x) - A^T y - \mu X^{-1} e \\ Ax - b \end{bmatrix}, \tag{6.6}$$

and recover Δz by $\Delta z = -z + \mu X^{-1} e - X^{-1} Z \Delta x$. A disadvantage of this system is that the term $J(x)^T J(x)$, hidden in $H(x)$, can be relatively dense. To avoid this difficulty, we propose defining the auxiliary variables $\xi := J(x)\Delta x$, and we need to solve the following larger but sparse system

$$\begin{bmatrix} B(x) + X^{-1}Z & A^T & J(x)^T \\ A & 0 & 0 \\ J(x) & 0 & -I \end{bmatrix} \begin{bmatrix} \Delta x \\ -\Delta y \\ \xi \end{bmatrix} = - \begin{bmatrix} J(x)^T F(x) - A^T y - \mu X^{-1} e \\ Ax - b \\ 0 \end{bmatrix}, \tag{6.7}$$

where $B(x)$ represents $\sum_{i=1}^m f_i(x) \nabla^2 f_i(x)$ or a symmetric to it. A second advantage of the above approach is that the system (6.7) is always invertible if A is full rank, even if $J(x)$ is not full rank. We propose solving (6.7) via a *regularization* of this system, which is in the very simple form

$$\begin{bmatrix} B(x) + X^{-1}Z + \rho I & A^T & J(x)^T \\ A & -\delta I & 0 \\ J(x) & 0 & -I \end{bmatrix} \begin{bmatrix} \Delta x \\ -\Delta y \\ \xi \end{bmatrix} = - \begin{bmatrix} J(x)^T F(x) - A^T y - \mu X^{-1} e \\ Ax - b \\ 0 \end{bmatrix}, \tag{6.8}$$

where $\rho > 0$ and $\delta > 0$ are regularization parameters. They affect only the coefficient matrix of the system and not the right hand side. These parameters make the system (6.8) invertible independent of A and $J(x)$. The role of the parameter δ is to regularize the constraints that may be (nearly) dependent, while the role of ρ is to regularize the $(1,1)$ block in the case where some variables are not subject to a non negativity constraint and/or in the case of a linear least-squares problem i.e., $B(x) = 0$.

Under these conditions, it is natural to assume that for sufficiently small values of ρ and δ, the solutions of (6.5) and (6.8) are close and that we can find a solution of (6.3).

6.2.3 Theoretical Aspects

We believe that the regularization (6.8) is justified by a mixed proximal point and augmented Lagrangian applied to (6.3). Indeed, by applying the augmented Lagrangian method to (6.3) we get

$$\min_{x} \ \tfrac{1}{2}\sum_{i=1}^{m} f_i(x)^2 - y_k^T(Ax-b) + \tfrac{1}{2\delta}\|Ax-b\|^2 \quad \text{subject to } x \geq 0,$$

which can also be written as:

$$\min_{x,r} \ \tfrac{1}{2}\sum_{i=1}^{m} f_i(x)^2 + \tfrac{1}{2}\delta\|r+y_k\|^2 \quad \text{subject to } Ax+\delta r = 0,\ x \geq 0.$$

We propose adding a proximal type term to this sub-problem in the spirit of Friedlander and Orban (2012) :

$$\min_{x,r} \ \tfrac{1}{2}\sum_{i=1}^{m} f_i(x)^2 + \tfrac{1}{2}\delta\|r+y_k\|^2 + \tfrac{1}{2}\rho\|x-x_k\|^2 \quad \text{subject to } Ax+\delta r = 0,\ x \geq 0. \tag{6.9}$$

Applying Newton's method to the Lagrangian of (6.9) as before, we obtain the Newton system

$$\begin{bmatrix} H(x)+\rho I & 0 & -A^T & -I \\ 0 & \delta I & -\delta I & 0 \\ A & \delta I & 0 & 0 \\ Z & 0 & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} J(x)^T F(x) + \rho(x-x_k) - A^T y - z \\ \delta(r+y_k) - \delta y \\ Ax + \delta r - b \\ Xz - \mu e \end{bmatrix}.$$

Eliminating Δr, we find the system

$$\begin{bmatrix} H(x)+\rho I & A^T & -I \\ A & -\delta I & 0 \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ -\Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} J(x)^T F(x) + \rho(x-x_k) - A^T y - z \\ Ax + \delta(y-y_k) - b \\ Xz - \mu e \end{bmatrix}. \tag{6.10}$$

Rather than using (6.10) directly, we propose the following symmetrization, obtained via the

similarity transformation defined by the diagonal matrix blkdiag$(I, I, I, Z^{-\frac{1}{2}})$

$$\begin{bmatrix} B & A^T & J(x)^T & Z^{\frac{1}{2}} \\ A & -\delta I & & \\ J(x) & & -I & \\ -Z^{\frac{1}{2}} & & & -X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta r \\ \Delta y \\ Z_k^{-\frac{1}{2}} \Delta z \end{bmatrix} = \begin{bmatrix} c - B^T y_k - A^T \lambda_k - z_k \\ d - A x_k - r_k \\ b - B x_k \\ \tau_k Z_k^{-\frac{1}{2}} e - X_k Z_k^{\frac{1}{2}} e \end{bmatrix}, \tag{6.11}$$

the advantages of (6.11) is that the linear systems used in the definition of the Newton steps are larger, sparser and tailored to the special structure of (6.9).

REFERENCES

Patrick R. Amestoy, Timothy A. Davis, and Iain S. Duff. An approximate minimum degree ordering algorithm. In *SIAM Journal on Matrix Analysis and Applications*, volume 30, pages 381—388, New York, NY, USA, sep, 2004. ACM. DOI: 10.1145/1024074.1024081.

M. Arioli and D. Orban. Iterative methods for symmetric quasi-definite linear systems. Cahier du GERAD G-2012-xx, GERAD, Montréal, Québec, Canada, 2012. In preparation.

P. Armand and J. Benoist. Uniform boundedness of the inverse of a Jacobian matrix arising in regularized interior-point methods. *Mathematical Programming*, 2011. DOI: 10.1007/s10107-011-0498-3.

J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–148, 1988. DOI: 10.1093/imanum/8.1.141.

Anne-Marie Baudron and Jean-Jacques Lautard. Minos: a simplified pn solver for core calculation. *Nuclear science and engineering*, 155(2):250–263, 2007.

S. Bellavia, M. Macconi, and B. Morini. An interior point newton-like method for non-negatiev least-squares problems with degenerate solution. *Numerical Linear Algebra with Applications*, 13:825–846, 2006. DOI: 10.1002/nla.502.

S. Bellavia, J. Gondzio, and B. Morini. Regularization and preconditioning of KKT systems arising in non-negative least-squares problems. *Numerical Linear Algebra with Applications*, 16:39–61, 2008. DOI: 10.1002/nla.610.

S. Bellavia, J. Gondzio, and B. Morini. Computational experience with numerical methods for nonnegative least-squares problems. *Numerical Linear Algebra with Applications*, 18: 363–385, 2011. DOI: 10.1002/nla.732.

M. Bierlaire, Ph. L. Toint, and D. Tuyttens. On iterative algorithms for linear least squares problems with bound constraints. *Linear Algebra and its Applications*, 143:111–143, 1991. DOI: 10.1016/0024-3795(91)90009-L.

Å. Bjorck. *Numerical Methods for Least Squares Problems*. Number OT51. SIAM, Philadelphia, PA, 1996. DOI: 10.1137/1.9781611971484.

S.S. Chen, D.L. Donoho, and M.A. Saunders. Atomic decomposition by basis pursuit. *SIAM review*, pages 129–159, 2001. DOI: 10.1137/S003614450037906X.

ILOG CPLEX. 11.0 users manual. *ILOG CPLEX Division. Incline Village, NV*, 2007.

B.N. Datta. *Numerical linear algebra and applications*. Society for Industrial Mathematics, 2010.

E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. DOI: 10.1007/s101070100263.

I. S. Duff. MA57—a code for the solution of sparse symmetric definite and indefinite systems. *Transactions of the ACM on Mathematical Software*, 30(2):118–144, 2004. DOI: 10.1145/992200.992202.

M. P. Friedlander. BCLS: Bound constrained least squares, 2007. URL `www.cs.ubc.ca/~mpf/bcls`. Accessed on 07/27/2012.

M. P. Friedlander and D. Orban. A primal-dual regularized interior-point method for convex quadratic programs. *Mathematical Programming Computation*, 4(1):71–107, 2012. DOI: 10.1007/s12532-012-0035-2.

P. E. Gill, M. A. Saunders, and J. R. Shinnerl. On the stability of Cholesky factorization for symmetric quasidefinite systems. *SIAM Journal on Matrix Analysis and Applications*, 17(1):35–46, 1996. DOI: 10.1137/S0895479893252623.

Philip E Gill, Walter Murray, and Michael A Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005. DOI: 10.1137/S0036144504446096.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Series in the Mathematical Sciences. Johns Hopkins, Baltimore, third edition, 1996.

N. I. M. Gould. On the accurate determination of search directions for simple differentiable penalty functions. *IMA Journal of Numerical Analysis*, 6:357–372, 1986. DOI: 10.1093/imanum/6.3.357.

C. Greif, E. Moulding, and D. Orban. Bounds on the eigenvalues of block matrices arising from interior-point methods. Cahier du GERAD G-2012-42, GERAD, Montréal, Québec, Canada, 2012.

R.J. Hanson and C.L. Lawson. Extensions and applications of the householder algorithm for solving linear least squares problems. *Mathematics of Computation*, pages 787–812, 1969.

HSL. *The HSL Mathematical Software Library*. STFC Rutherford Appleton Laboratory, 2007. http://www.hsl.rl.ac.uk.

C.A. Johnson, J. Seidel, and A. Sofer. Interior-point methodology for 3-d pet reconstruction. *Medical Imaging, IEEE Transactions on*, 19(4):271–285, 2000.

S. J. Kim, K. Koh, M. Lusting, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE Journal on Selected Topics in Signal Processing*, 1(4):606–617, 2007a. DOI: 10.1109/JSTSP.2007.910971.

S.J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *Selected Topics in Signal Processing, IEEE Journal of Machine learning research*, 1(4):606–617, 2007b. DOI: http://doi.ieeecomputersociety.org/10.1109/TVCG.2011.157.

M Kojima, N Megiddo, and S Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Mathematical programming*, 61(1):263–280, 1993.

C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems*. Number 15 in Classics in Applied Mathematics. SIAM, Philadelphia, PA, 1995. DOI: 10.1137/1.9781611971217. Republication of original 1974 edition.

K. Levenberg. A method for the solution of certain problems in least squares. *Quarterly of applied mathematics*, 2:164–168, 1944.

J. Nocedal and S.J. Wright. *Numerical optimization*. Springer verlag, 1999.

D. Orban. NLPy—a large-scale optimization toolkit in Python. Technical Report Cahier du GERAD G-2012-xx, GERAD, 2012. https://github.com/dpo/nlpy.

C.C. Paige and M.A. Saunders. Lsqr: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982. DOI: 10.1145/355984.355989.

L. F. Portugal, J. J. Judice, and L. N. Vicente. Comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables. *Mathematics of Computation*, 63:625–643, 1994. DOI: 10.1090/S0025-5718-1994-1250776-4.

R. T. Rockafellar. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1:97–116, 1976. DOI: 10.1287/moor.1.2.97.

D. Silvester and A. Wathen. Fast iterative solution of stabilised stokes systems part ii: using general block preconditioners. *SIAM Journal on Numerical Analysis*, pages 1352–1367, 1994. DOI: http://dx.doi.org/10.1137/0731070.

L.N. Trefethen and D. Bau. *Numerical linear algebra*. Number 50. Society for Industrial Mathematics, Philadelphia, 1997.

R. J. Vanderbei. Symmetric quasi-definite matrices. *SIAM Journal on Optimization*, 5(1): 100–113, 1995. DOI: 10.1137/0805005.

R J Vanderbei. Loqo: An interior point code for quadratic programming. *Optimization methods and software*, 11(1-4):451–484, 1999.

Andreas Waechter. Introduction to ipopt, 2007.

R Waltz and J Nocedal. Knitro 2.0 users manual. Technical report, Technical Report OTC 2003, 2003.

S.J. Wright. *Primal-dual interior-point methods*, volume 54. Society for Industrial Mathematics, Philadelphia, PA, 1997.

ANNEXE A : IMPLEMENTATION

Until now, we have considered our problems to be given in standard form. However, for real-world problems it is often convenient to formulate problems in the following form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & c^T x + \tfrac{1}{2}\|Ax - d\|^2 \\ \text{subject to} \quad & L \leq Bx \leq U, \\ & l \leq x \leq u. \end{aligned} \tag{6.12}$$

Two-sided constraints such as those given here are called constraints with ranges. The vector L is called the vector of lower bound constraints, U is the vector of upper bound constraints, l is called the vector of lower bound variables, and u is the vector of upper bound variables. We allow some of the data to take infinite values, that is, for each $i = 1, 2, \cdots, m$

$$-\infty \leq L_i \leq U_i \leq \infty,$$

and for each $i = 1, 2, \cdots, n$

$$-\infty \leq l_i \leq u_i \leq \infty.$$

We have implemented the primal-dual regularization methods described in Algorithm (Friedlander and Orban, 2012, Algorithm 4.1) and specialized in Algorithms 3.1.1 in the Python programming language as part of the NLPy programming platform for optimization (Orban, 2012). Our implementation handles least-squares problem in the slack formulation form of (6.12), i.e., the primal-dual pair

$$\begin{aligned} \underset{x,r,t}{\text{minimize}} \quad & c^T x + \tfrac{1}{2}\|r\|^2 \\ \text{subject to} \quad & Bx + B_1 t = b, \quad Ax + r = d, \quad t \geq 0, \end{aligned}$$

and

$$\begin{aligned} \underset{x,r,y,z}{\text{maximize}} \quad & b^T y - (A^T d)^T x - \tfrac{1}{2}\|r\|^2 \\ \text{subject to} \quad & B^T y - A^T A x = c - A^T d, \; B_1^T y + z = 0, \; z \geq 0, \end{aligned}$$

where $c \in \mathbb{R}^n, A \in \mathbb{R}^{p \times n}, B \in \mathbb{R}^{m \times n}, B_1 \in \mathbb{R}^{m \times q}, b \in \mathbb{R}^m, d \in \mathbb{R}^p, x \in \mathbb{R}^n, r \in \mathbb{R}^n, t \in \mathbb{R}^q, y \in \mathbb{R}^m$, and $z \in \mathbb{R}^n$. We consider two approaches to compute the Newton step at iteration k. In this thesis, we work with a 4×4 system, which is obtained by eliminating ΔZ from Newton

system, i.e.,

$$\begin{bmatrix} -\rho I & & A^T & B^T \\ & -\rho I & & B_1^T \\ A & & I & \\ B & B_1 & & \delta I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta t \\ \Delta r \\ \Delta y \end{bmatrix} = \begin{bmatrix} c - A^T r - B^T y \\ -B_1^T y - z \\ d - Ax - r \\ b - Bx - B_1 t \end{bmatrix}.$$

In Chapter 4, we consider the following 5×5 system

$$\begin{bmatrix} -\rho I & & A^T & B^T & \\ & -\rho I & & B_1^T & Z^{\frac{1}{2}} \\ A & & I & & \\ B & B_1 & & \delta I & \\ & Z^{\frac{1}{2}} & & & T \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta t \\ \Delta r \\ \Delta y \\ Z^{-\frac{1}{2}} \Delta z \end{bmatrix} = \begin{bmatrix} c - A^T r - B^T y \\ -B_1^T y - z \\ d - Ax - r \\ b - Bx - B_1 t \\ Z^{-\frac{1}{2}} (\sigma_k \mu_k e - T_k z_k) \end{bmatrix}.$$

In this case, we have a sparser system and the LDL^T factorization performs better.

ANNEXE B : NUMERICAL ALGEBRA REVIEW

In this section, we discuss implementation issues that arise in solving linear systems. The most important issue is to find a way to solve the systems of equations discussed in the previous section. Factoring a positive definite matrix is the main part of solving a linear system. We shall study factorization techniques for the reduced KKT matrix. We factor square positive definite matrices into lower, upper, and diagonal components. The LDL^T Factorization factors the square matrix K as

$$K = LDL^T. \tag{6.13}$$

Every symmetric and positive definite matrix K can be written as (6.13), where L is a lower triangular matrix with unit diagonal elements and D is a diagonal matrix with positive elements on the diagonal. By equating the elements in (6.13), column by column, it is easy to derive formulas for computing L and D. The LDL^T factorization for indefinite matrices may not exist. In a general case, we call LDL^T factorization as symmetric indefinite factorization

$$PKP^T = LDL^T, \tag{6.14}$$

where P is a permutation matrix, L is unit lower triangular, and D is block diagonal with diagonal blocks of diminution 1 or 2. The second factorization is produced by Aasen's method

$$PKP^T = LTL^T, \tag{6.15}$$

where P is a permutation matrix, L is unit lower triangular, and T is unit lower triangular with first column e_1. This kind of factorization is much less widely used than block LDL^T factorization, but it is mathematically interesting. The reduced KKT system is an example of a symmetric quasi-definite system.

ANNEXE C : THE APPROXIMATE MINIMUM DEGREE (AMD)

When solving large sparse symmetric linear systems of the form $Kx = r$, it is common to precede the numerical factorization by a symmetric reordering. This reordering is chosen so that pivoting down the diagonal in order on the resulting permuted matrix $PKP^T = LL^T$ produces much less fill-in and less work than computing the factors of K by pivoting down the diagonal in the original order. The goal of the preordering is to find a permutation matrix P so that the subsequent factorization has the least fill-in. Unfortunately, this problem is NP-complete, so heuristics are used. For more details, see, for example, Amestoy et al. (2004). The minimum degree ordering algorithm is one of the most widely used heuristics, since it produces factors with relatively low fill-in on a wide range of matrices. In python, the following command can be used

commentstyle

```
>>> from cvxopt import spmatrix, amd
>>> A = spmatrix([10,3,5,-2,5,2], [0,2,1,2,2,3], [0,0,1,1,2,3])
>>> P = amd.order(A)
>>> print(P)
[ 1]
[ 0]
[ 2]
[ 3]
```

In MATLAB, we can use the *amd* command.

ANNEXE D : SQD MATRIX AND LDL^T FACTORIZATION

A symmetric quasi-definite system is a special case of systems that arise in interior point methods when we are searching for the new direction by solving the Newton system. Coefficient matrix in such a system is closely related to a symmetric positive-definite matrix called symmetric quasi-definite (SQD) matrix. The advantages of the SQD matrix is that it is nonsingular, and that its inverse is again symmetric quasi-definite. We can extend the facts of positive definite matrix to this kind of matrix. Here, we define SQD matrix and we shall show how LDL^T factorization can be used for this kind of matrix.

Theorem 6.2.1. *Let matrix K have the following form*

$$\begin{bmatrix} A & B \\ D & C \end{bmatrix},$$

where A , B, C, and D are matrices and A is positive definite.
K is positive definite, if and only if $C - DA^{-1}B$ is positive definite.

Proof. Using the fact that K is positive definite implies that

$$\begin{bmatrix} x^T & y^T \end{bmatrix} \begin{bmatrix} A & B \\ D & C \end{bmatrix} \begin{bmatrix} x & y \end{bmatrix} = x^T Ax + x^T By + y^T Dx + y^T Cy > 0.$$

Fix a vector $y \neq 0$, and put $x = -A^{-1}By$. Using this choice in the last equation, we get

$$y^T B^T A^{-T} AA^{-1} By - y^T B^T A^{-1} By - y^T DA^{-1} By + y^T Cy = y^T (C - DA^{-1}B)y > 0.$$

Since y is an arbitrary nonzero vector, it follows that $C - DA^{-1}B$ is positive definite. Now, suppose that $C - DA^{-1}B$ is positive definite since we have that

$$K = \begin{bmatrix} A & B \\ D & C \end{bmatrix} = \begin{bmatrix} I & 0 \\ DA^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & C - DA^{-1}B \end{bmatrix} \begin{bmatrix} I & A^{-1}B \\ 0 & I \end{bmatrix}, \tag{6.16}$$

the eigenvalues of K are the eigenvalues of the A and its Schur complement, which are therefore positive. □

Corollary 6.2.2. *Let matrix K have the following form*

$$\begin{bmatrix} a & b^T \\ b & C \end{bmatrix},$$

where a is a scaler and b is a vector and C is a symmetric matrix.
If K is positive definite, then so is $C - \frac{bb^T}{a}$.

Proof. The proof is an immediate application of theorem 6.2.1 □

Definition 6.2.3. *A symmetric matrix is called quasi-definite if it can be written perhaps after a symmetric permutation as*

$$\begin{bmatrix} -E & A \\ A^T & D \end{bmatrix},$$

where E and D are positive definite matrices.

Quasi-definite matrices inherit some of the nice properties of positive definite matrices. In fact, any arbitrary symmetric permutation of rows or columns gives us a factorization of a permuted matrix. When we perform an arbitrary symmetric permutation and elimination, the fact is that the remaining uneliminated part of the whole matrix is still quasi-definite. Below, we illustrate the justification of this fact.

Let us break out the first row/column of the matrix and look at the first step of the elimination process. Breaking out the first row/column of K, we write

$$\begin{bmatrix} -a & -b^T & f^T \\ -b & -C & G \\ f & G^T & D \end{bmatrix},$$

where a is a scalar, b and f are vectors, and C, D, and G are symmetric matrices of the appropriate dimensions. One step of the elimination process transforms K into

$$\begin{bmatrix} 1 & \frac{-b^T}{a} & \frac{f^T}{a} \\ 0 & -(C - \frac{bb^T}{a}) & G + \frac{bf^T}{a} \\ 0 & G^T + \frac{fb^T}{a} & D + \frac{ff^T}{a} \end{bmatrix},$$

the uneliminated part is

$$\begin{bmatrix} -(C - \frac{bb^T}{a}) & G + \frac{bf^T}{a} \\ G^T + \frac{fb^T}{a} & D + \frac{ff^T}{a} \end{bmatrix}.$$

Clearly, the lower-left and upper-right blocks are transposes of each other. Also, the upper-left and lower-right blocks are symmetric. Therefore, the whole matrix is symmetric. The theorem 6.2.2 tells us $C-\frac{bb^T}{a}$ and $D+\frac{ff^T}{a}$ are positive definite. Since we have the fact that the sum of a positive definite matrix and a positive semidefinite matrix is positive definite, the uneliminated part is indeed quasi-definite. Hence, it is clear that the uneliminated part is also quasi-definite no matter which diagonal element is selected as the first pivot element. Every step of the elimination process needs to choose a pivot element from the diagonals of the matrix. Since these diagonals come from either a positive definite sub matrix or the negative of such a matrix, we can say that they are nonzero but many of them will be negative. Therefore, for any positive definite matrix, an arbitrary symmetric permutation of such a matrix can be factored without any difficulty of encountering a zero pivot element. Hence, after one step of the elimination, the uneliminated part is positive definite. It therefore follows by induction that the uneliminated part is positive definite at every step of the elimination. We implemented the procedure called *s*qd_pivoit.m in MATLAB. The following is a simple example: let

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & -1 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & -1 & -1 & 3 & -1 \\ -1 & 0 & 0 & -1 & 3 \end{bmatrix},$$

at the end of the four steps of the elimination without permutations, we end up with the lower triangular matrix L with ones on the diagonal and the diagonal matrix D. It is convenient to combine the lower triangular matrix with the diagonal matrix to get a new lower triangular matrix with ones on the diagonal. But the current L is exactly the transpose of the upper triangular matrix. Hence, to preserve symmetry, we should combine the diagonal matrix with both the lower and the upper triangular matrices to have an LDL^T factorization of A where

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1/2 & 1 & 0 & 0 & 0 \\ 0 & -2/5 & 1 & 0 & 0 \\ 0 & -2/5 & -7/8 & 1 & 0 \\ -1/2 & -1/5 & -1/8 & -1 & 1 \end{bmatrix}$$

and

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 5/2 & 0 & 0 & 0 \\ 0 & 0 & 8/5 & 0 & 0 \\ 0 & 0 & 0 & 11/8 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Of course, once a factorization is found, it is easy to solve systems of equations using forward and backward substitution. The following theorem states a nice feature of SQD matrices. This property can be used to have sparse L in LDL^T factorization of a SQD matrix.

Theorem 6.2.4. *Any symmetric permutation of a SQD yields a factorization $PKP^T = LDL^T$, where L is unit lower triangular matrix and D is diagonal.*

Proof. See Vanderbei (1995). □

If K is symmetric indefinite there might not exist a permutation matrix P such that $PKP^T = LDL^T$ for some L unit lower triangular and diagonal D. As a simple example, let

$$K = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

It is easy to see that K is not SQD and dos not have LDL^T factorization. It is also obvious that for any permutation matrix P we have $PKP^T = K$. Even when there exists such P we may have a numerical stability problem. Example:

$$\begin{bmatrix} \epsilon & 1 \\ 1 & \epsilon \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 0 \\ 0 & \epsilon - 1/\epsilon \end{bmatrix} \begin{bmatrix} 1 & 1/\epsilon \\ 1 & 0 & 1 \end{bmatrix}.$$

The Cholesky factorization is LDL^T with $D = I$. Another fact about SQD matrices is that they are indefinite and non-singular. To see this, consider the following system.

$$\begin{bmatrix} -E & A^T \\ A & F \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}.$$

The positive definiteness of E allows us to solve the first set of equations for x in terms of y and obtain

$$x = E^{-1}(b - A^T y),$$

substituting x into the second set of equations yields

$$y = (F + AE^{-1}A^T)(c + AE^{-1}b).$$

The positive definiteness of E and F ensures that $F + AE^{-1}A^T$ is positive definite. Hence, there exists a unique solution for any b and c, which implies that the SQD matrix is nonsingular.

ANNEXE E : MATLAB CODE FOR LDL^T FACTORIZATION OF SQD

```
function [L,D,P,p,error]=sqd_pivot(A)
dim = size(A);
p = amd(A);
p = [1 3 4 2];
tmp = A(p,p);
for i=1:dim(1)-1
    tmp=pivot(tmp,i);
end
D = diag(diag(tmp));
U = triu(tmp);
U = sparse(diag(1./diag(U)))*U;
L = U';
B = L*D*L';
I = eye(dim(1));
q = 1:dim(1);
P = I(p,q);
error = sparse(A-P'*B*P);
end

function [P1,P2] =permut(A,i,j,p,q)
dim = size(A);
n = dim(1);
P=eye(n);
P(i,i)= 0;
P(p,p)= 0;
P(i,p)= 1
P(p,i)= 1;
P1 =P;
P=eye(n);
P(j,j)= 0;
P(q,q)= 0;
P(j,q)= 1;
P(q,j)= 1;
P2 =P;
end
```

```
function x=sqd_solve(A,b)
%Using the SQD decomposition to solve Ax = b and return x.
%We rewrite Ax = b as LDL'x = b and let Ux = y.
%First we solve Ly = b using forward substitution to get y.
%Next, using this, we solve Ux = y using backward substitution to get x.
%Example:
%A=creat_sqd(3,4);
%b=rand(7,1);
% x=sqd_solve(A,b)

[L,D] = sqd_pivot(A);

y= forward_sub(L,b);
x= back_sub(D*L',y);
```

```
15
16  end
17
18  function y= forward_sub(L,w)
19
20      if nargin ~= 2
21         error 'Only two inputs are required.'
22      end
23
24      if ~(isnumeric(L)&isnumeric(w))
25          error 'input must be numeric.'
26      end
27
28      [nRow,nCol]=size(w);
29      if nRow>1 & nCol>1
30          error 'w must be a vector not a matrix.'
31      end
32
33      [nRow,nCol]=size(L);
34      if nRow ~= nCol
35          error 'Matrix L must be square.'
36      end
37
38      if length(w) ~= nRow
39          error 'w length does not match L matrix dimension'
40      end
41
42      y=zeros(nRow,1);
43      y(1) = w(1)/L(1,1);
44
45      w=w(:);
46
47      for n=2:nRow
48          y(n)=( w(n) - L(n,1:n-1)*y(1:n-1) ) / L(n,n);
49      end
50
51  end
52
53  function x=back_sub(U,v)
54
55      if nargin ~= 2
56         error 'Only two inputs are required.'
57      end
58
59      if ~(isnumeric(U)&isnumeric(v))
60          error 'input must be numeric.'
61      end
62
63      [nRow,nCol]=size(v);
64      if nRow>1 & nCol>1
65          error 'v must be a vector not a matrix.'
66      end
67
68      [nRow,nCol]=size(U);
69      if nRow ~= nCol
```

```
70          error 'Matrix U must be square.'
71      end
72
73      if length(v) ~= nRow
74         error 'v length does not match U matrix dimension.'
75      end
76
77      x=zeros(nRow,1);
78      x(nRow)=v(nRow)/U(nRow,nRow);
79
80      v=v(:);
81
82      for n=(nRow-1):-1:1
83          x(n)=(v(n)-(U(n,n+1:end)*x(n+1:end))) / U(n,n);
84      end
85
86  end
```