ECE 573 Course Notes
Advanced Engineering Design by Optimization

Wu-Sheng Lu

Department of Electrical and Computer Engineering

University of Victoria

January 2019

COURSE OUTLINE

Instructor:

Dr. W.-S. Lu
Phone: 8692
E-mail: wslu@ece.uvic.ca
URL: www.ece.uvic.ca/~wslu

Office Hours:

Days: Wednesdays
Time: 14:40 – 16:40
Location: EOW 427

Lectures:

Section: A01/CRN 20934
Days: Tuesdays, Wednesdays, and Fridays
Time: 1:30 – 2:20 pm
Location: Elliott 162

Text:

Lecture Notes

Assessment:

Assignments	25%
Project	20%
Final Exam	55%

Due Dates for Assignments and Project Report:

There will be ten assignments with due dates to be given in class as well as in the course web site. You are required to submit *paper version* of the assignment no later than 4:00pm on the due day. You may choose one of the following to submit your assignment work:

- submit to the instructor in class.
- submit by sliding it under the door of EOW 427 into the office.

The project report is due on the same day as the final exam.

Prerequisites and Preparation

1. · ECE 403 or ECE 503 or an equivalent course is essential.
   · It is unlikely that you will encounter serious difficulties with the course material if you are familiar with multivariable calculus and linear algebra. The course notes include an appendix on linear algebra.
2. Download software CVX version 2.1 from the web site:

   http://cvxr.com/cvx/download/

   and a user manual from

   http://cvxr.com/cvx/doc/

   You are encouraged to install the software as soon as the course begins, go through the first 60 or so pages of the manual, and try the examples given in the manual.

Primary Reference

A large part of the course material is based on Boyd's book:

S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.

For a soft copy of the book, visit Boyd's web site.

Contents

Chapter 0 Unconstrained Optimization: A Review

0.1 The Problem

A general unconstrained optimization problem assumes the form

$$\underset{x \in E^n}{\text{minimize}} \quad f(\boldsymbol{x})$$

where $f(\boldsymbol{x})$ is a real-valued *objective* function. This is to say, we seek to find a point $\boldsymbol{x}^*$ in space E^n such that at least in a small vicinity centered around $\boldsymbol{x}^*$ the value of $f(\boldsymbol{x}^*)$ is smallest. In that case $\boldsymbol{x}^*$ is said to be a *local minimizer* of $f(\boldsymbol{x})$. If $f(\boldsymbol{x}^*)$ is smallest in the entire E^n, $\boldsymbol{x}^*$ is said to be a *global minimizer* of $f(\boldsymbol{x})$.

Typically, we assumed the objective function $f(\boldsymbol{x})$ has continuously differentiable second-order derivatives, namely, $f(\boldsymbol{x}) \in C^2$.

Notation

- Gradient of $f(\boldsymbol{x})$

$$\nabla f(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}$$

- Hessian of $f(\boldsymbol{x})$

$$\nabla^2 f(\boldsymbol{x}) = \nabla\left(\nabla^T f(x)\right) = \boldsymbol{H}(\boldsymbol{x}) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \cdots & \dfrac{\partial^2 f}{\partial x_n \partial x_1} \\ \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_n \partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_1 \partial x_n} & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- Since $\dfrac{\partial^2 f}{\partial x_i \partial x_j} = \dfrac{\partial^2 f}{\partial x_j \partial x_i}$, $\boldsymbol{H}(\boldsymbol{x})$ is an $n \times n$ *symmetric* matrix.

0.2 Taylor Expansion of $f(x)$

$$f(\boldsymbol{x}+\boldsymbol{\delta}) = f(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})^T \boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T \boldsymbol{H}(\boldsymbol{x})\boldsymbol{\delta} + O\left(\|\boldsymbol{\delta}\|^3\right)$$

• *Linear approximation* of $f(\boldsymbol{x}+\boldsymbol{\delta})$

$$f(\boldsymbol{x}+\boldsymbol{\delta}) \approx f(\boldsymbol{x})+\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{\delta}$$

• *Quadratic approximation* of $f(\boldsymbol{x}+\boldsymbol{\delta})$

$$f(\boldsymbol{x}+\boldsymbol{\delta}) \approx f(\boldsymbol{x})+\boldsymbol{g}(\boldsymbol{x})^T\boldsymbol{\delta}+\frac{1}{2}\boldsymbol{\delta}^T\boldsymbol{H}(\boldsymbol{x})\boldsymbol{\delta}$$

0.3 Necessary and Sufficient Conditions for Local Minima

0.3.1 *First-order necessary condition*

Theorem 0.1 ***First-order necessary condition for a minimum***

If $\boldsymbol{x}^*$ is local minimizer of $f(\boldsymbol{x})$, then

$$\nabla f(\boldsymbol{x}^*)=\boldsymbol{0} \tag{0.1}$$

Proof: From Taylor expansion of $f(\boldsymbol{x})$ in a neighborhood of $\boldsymbol{x}^*$, namely

$$f(\boldsymbol{x}^*+\boldsymbol{\delta})=f(\boldsymbol{x}^*)+\nabla f(\boldsymbol{x}^*)^T\boldsymbol{\delta}+O(\|\boldsymbol{\delta}\|^2)$$

and $\boldsymbol{x}^*$ being a local minimizer, we have

$$f(\boldsymbol{x}^*+\boldsymbol{\delta})-f(\boldsymbol{x}^*)=\nabla f(\boldsymbol{x}^*)^T\boldsymbol{\delta}+O(\|\boldsymbol{\delta}\|^2)\geq 0$$

which implies that, for *any* $\boldsymbol{\delta}$ of small magnitude, $\nabla f(\boldsymbol{x}^*)^T\boldsymbol{\delta}\geq 0$. By choosing $\boldsymbol{\delta}=-\nabla f(\boldsymbol{x}^*)$ (up to an appropriate scaling factor), this means that $-\|\nabla f(\boldsymbol{x}^*)\|^2\geq 0$ which immediately leads to Eq. (0.1). ■

Definition

A point $\boldsymbol{x}$ is said to be a *stationary point* of a smooth objective function $f(\boldsymbol{x})$ if $\nabla f(\boldsymbol{x})=\boldsymbol{0}$.

If follows from Theorem 0.1 that if $\boldsymbol{x}^*$ is a local minimizer of $f(\boldsymbol{x})$, then $\boldsymbol{x}^*$ must be a stationary point of $f(\boldsymbol{x})$.

0.3.2 *Second-order necessary conditions*

Definition

(*a*) Let $\boldsymbol{d}$ be an arbitrary direction vector at point $\boldsymbol{x}$. The quadratic form $\boldsymbol{d}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{d}$ is said to be *positive definite, positive semidefinite, negative semidefinite, negative definite* if $\boldsymbol{d}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{d}$ > 0, ≥ 0, ≤ 0, < 0, respectively, for all $\boldsymbol{d}\neq\boldsymbol{0}$ at $\boldsymbol{x}$. If $\boldsymbol{d}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{d}$ can assume positive as well as negative values, it is said to be *indefinite*.

(*b*) If $\boldsymbol{d}^T\nabla^2 f(\boldsymbol{x})\boldsymbol{d}$ is positive definite positive semidefinite, etc., then matrix $\nabla^2 f(\boldsymbol{x})$ is said to be positive definite (denoted by $\nabla^2 f(\boldsymbol{x})\succ\boldsymbol{0}$), positive semidefinite (denoted by $\nabla^2 f(\boldsymbol{x})\succeq\boldsymbol{0}$), etc.

Theorem 0.2 A symmetric matrix is positive definite, positive semidefinite, negative semidefinite, negative definite, or negative semidefinite if and only if its engenvalues are positive, non-negative, negative, or non-positive, respectively.

Theorem 0.3 ***Second-order necessary conditions for a minimum***

If $f(\boldsymbol{x}) \in C^2$ and $\boldsymbol{x}^*$ is a local minimizer, then

(a) $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$

(b) $\nabla^2 f(\boldsymbol{x}^*)$ is positive semidefinite.

Proof From Taylor expansion of $f(\boldsymbol{x})$ in a neighborhood of $\boldsymbol{x}^*$, namely

$$f(\boldsymbol{x}^* + \boldsymbol{\delta}) = f(\boldsymbol{x}^*) + \nabla f(\boldsymbol{x}^*)^T \boldsymbol{\delta} + \tfrac{1}{2}\boldsymbol{\delta}^T \nabla^2 f(\boldsymbol{x}^*)^T \boldsymbol{\delta} + O(\|\boldsymbol{\delta}\|^3)$$

in conjunction with Eq. (0.1) and $\boldsymbol{x}^*$ being a local minimizer, we have

$$f(\boldsymbol{x}^* + \boldsymbol{\delta}) - f(\boldsymbol{x}^*) = \tfrac{1}{2}\boldsymbol{\delta}^T \nabla^2 f(\boldsymbol{x}^*)^T \boldsymbol{\delta} + O(\|\boldsymbol{\delta}\|^3) \ge 0$$

which implies that, for *any* $\boldsymbol{\delta}$ of small magnitude, $\boldsymbol{\delta}^T \nabla^2 f(\boldsymbol{x}^*)^T \boldsymbol{\delta} \ge 0$. Therefore $\nabla^2 f(\boldsymbol{x}^*)$ is positive semidefinite. ■

The example below shows that the conditions $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$ and $\nabla^2 f(\boldsymbol{x}^*) \succeq \mathbf{0}$ are *not enough* for $\boldsymbol{x}^*$ to be a minimizer.

Example 0.1 Consider the function

$$f(x_1, x_2) = x_1^2 - 2x_1 + x_2^3$$

By setting the gradient $\nabla f(\boldsymbol{x})$ to zero, we have

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} 2x_1 - 2 \\ 3x_2^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

which yields the only candidate point as

$$\boldsymbol{x}^* = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

At $\boldsymbol{x} = \boldsymbol{x}^*$, the Hessian is found to be

$$\nabla^2 f(\boldsymbol{x}^*) = \begin{bmatrix} 2 & 0 \\ 0 & 6x_2 \end{bmatrix}\Bigg|_{\boldsymbol{x}=\boldsymbol{x}^*} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

which is obviously positive semidefinite (but *not* positive definite). Hence point $\boldsymbol{x}^*$ satisfies the second-order necessary conditions.

The question now is whether $\boldsymbol{x}^*$ is a local minimizer of $f(\boldsymbol{x})$?

To address the question, we examine the difference $f(\boldsymbol{x}^* + \boldsymbol{\delta}) - f(\boldsymbol{x}^*)$ as follows:

$$f(\boldsymbol{x}^* + \boldsymbol{\delta}) - f(\boldsymbol{x}^*) = \left(\delta_1^2 + \delta_2^3 - 1\right) - (-1) = \delta_1^2 + \delta_2^3$$

If we draw a line that vertically passes though point $\boldsymbol{x}^* = [1 \quad 0]^T$ and move a point along that line crossing point $\boldsymbol{x}^*$ while evaluating how the value of function $f(\boldsymbol{x})$ changes relative to the value of $f(\boldsymbol{x}^*) = -1$, we are actually evaluating the difference $f(\boldsymbol{x}^* + \boldsymbol{\delta}) - f(\boldsymbol{x}^*)$: when the point on the vertical line moves from $\boldsymbol{x}^*$ to the right, we have $\delta_1 = 0$ and $\delta_2 > 0$, hence $f(\boldsymbol{x}^* + \boldsymbol{\delta})$ grows up (thus $\boldsymbol{x}^*$ can't be a maximizer); and when the point on the vertical line moves from $\boldsymbol{x}^*$ to the left, we have $\delta_1 = 0$ and $\delta_2 < 0$, hence $f(\boldsymbol{x}^* + \boldsymbol{\delta})$ goes down (thus $\boldsymbol{x}^*$ can't be a minimizer). Therefore we conclude that $\boldsymbol{x}^*$ is a saddle point. The function in a small neighborhood of $\boldsymbol{x}^*$ is depicted in Fig. 0.1. ■

Theorem 0.4 ***Second-order sufficient conditions for a minimum***

Suppose $f(\boldsymbol{x}) \in C^2$. Then the conditions

(a) $\nabla f(\boldsymbol{x}^*) = \mathbf{0}$

(b) $\nabla^2 f(\boldsymbol{x}^*)$ is positive definite

are sufficient for $\boldsymbol{x}^*$ to be a strong local minimizer.

Proof With an argument almost identical to that in Theorem 0.2, under conditions (a) and (b) we have for sufficiently small $\boldsymbol{\delta}$

$$f(\boldsymbol{x}^* + \boldsymbol{\delta}) - f(\boldsymbol{x}^*) = \tfrac{1}{2}\boldsymbol{\delta}^T \nabla^2 f(\boldsymbol{x}^*)^T \boldsymbol{\delta} + O(\|\boldsymbol{\delta}\|^3) > 0$$

Hence $f(\boldsymbol{x}^* + \boldsymbol{\delta}) > f(\boldsymbol{x}^*)$. ■

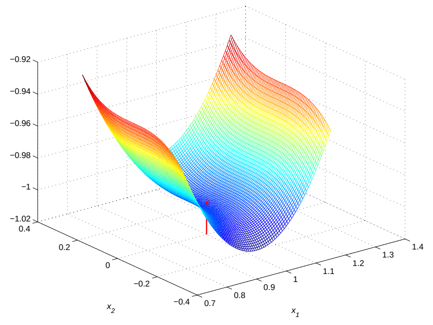


Figure 0.1 Function $f(\boldsymbol{x})$ in a small neighborhood of $\boldsymbol{x}^*$ for Example 0.1.

0.4 General Structure of Optimization Algorithms

Most of the available optimization algorithms entail a series of steps which are executed sequentially. A typical pattern is as follows:

Step 1 Input an initial point $\boldsymbol{x}_0$ and a convergence tolerance ε, set $k = 0$.

Step 2 Given $\boldsymbol{x}_k$, compute a search direction $\boldsymbol{d}_k$ by an appropriate procedure.

Step 3 Compute a positive scalar α_k that minimizes $f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$. Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$.

Step 4 If $\|\alpha_k \boldsymbol{d}_k\| < \varepsilon$, output $\boldsymbol{x}_{k+1}$ as the solution and stop; otherwise, set $k := k+1$ and go to Step 2.

0.5 Gradient Descent (GD) Method

The GD algorithm is a popular algorithm that iteratively secures a local minimizer by using the gradient of the objective function. The method is based on the observation that the value of a smooth $f(\boldsymbol{x})$ increases most rapidly along the direction of its gradient $\nabla f(\boldsymbol{x})$. This is because

$$f(\boldsymbol{x}+\boldsymbol{\delta})-f(\boldsymbol{x}) \approx \nabla f(\boldsymbol{x})^T \boldsymbol{\delta} = \|\nabla f(\boldsymbol{x})\| \cdot \|\boldsymbol{\delta}\| \cdot \cos\left\langle \nabla f(\boldsymbol{x}), \boldsymbol{\delta} \right\rangle$$

where $\cos\left\langle \nabla f(\boldsymbol{x}), \boldsymbol{\delta} \right\rangle$ reaches its largest value 1 when $\boldsymbol{\delta}$ is set to be $\nabla f(\boldsymbol{x})$.

In other words, $-\nabla f(\boldsymbol{x})$ is the gradient descent direction to reduce the objective, and this suggests the well-known formula that updates a current iterate $\boldsymbol{x}_k$ to the next iterate $\boldsymbol{x}_{k+1}$ as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k) \tag{0.2}$$

where $\alpha_k > 0$ is a positive real that controls the step size of the iteration. The process of determining a suitable α_k is called *line search*. Below we present a simple yet effective line search method that employs a *back-tracking* strategy.

Backtracking Line Search (BLS)

Input: Given $\boldsymbol{x}_k \in \text{dom}(f)$ and a descent direction $\boldsymbol{d}_k$ of $f(\boldsymbol{x})$ at $\boldsymbol{x}_k$.

Step 1 Select constants $\rho \in (0, 0.5)$ and $\gamma \in (0,1)$. Set $\alpha = 1$.

Step 2 While $f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) > f(\boldsymbol{x}_k) + \rho\alpha \boldsymbol{g}_k^T \boldsymbol{d}_k$, set $\alpha := \gamma\alpha$.

Step 3 Output $\alpha_k = \alpha$.

To explain how the BLS works, note that with a constant $\rho \in (0, 0.5)$ the expression on the right-hand side of the inequality in Step 2 represents a line in the coordinate system for function of α versus α that passes through the point $(0, f(\boldsymbol{x}_k))$ with a negative slop because

$$\rho \boldsymbol{g}_k^T \boldsymbol{d}_k = \rho \left. \frac{df(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)}{d\alpha} \right|_{\alpha=0} < 0$$

see Fig. 0.2. From the figure, we observe that a value of α less than α_0 may be deemed acceptable because such an α insures that $f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$ is no greater than $f(\boldsymbol{x}_k) + \rho\alpha \boldsymbol{g}_k^T \boldsymbol{d}_k$. This explains Step 2 which says one shall keep reducing the value of α by a factor of γ until the inequality in Step 2 is no longer valid. The MATLAB code of this line search method is named as bt_lsearch.m and is available from the course website.

Remark

Below is MATLAB pseudocode that can be useful to estimate average CPU time required by an algorithm:

1. Set `Initial_CPU_time = cputime`;
2. `for i = 1:1000, run Algorithm 5.1; end`
3. Set `Final_CPU_time = cputime;`
4. Compute `Average CPU time = (Final_CPU_time - Initial_CPU_time)/1000;`

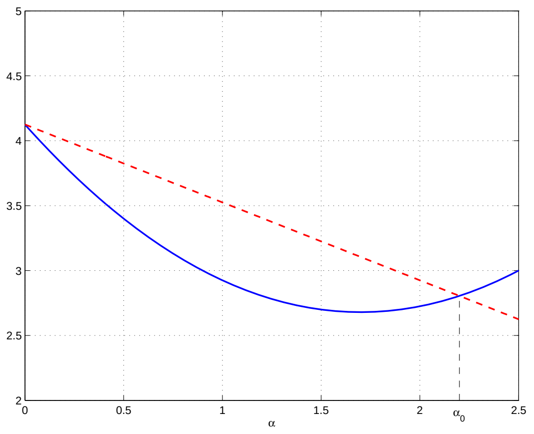


Fig. 0.2 Both $f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$ (blue curve) and $f(\boldsymbol{x}_k) + \rho\alpha \boldsymbol{g}_k^T \boldsymbol{d}_k$ (red dashed line) are shown as functions of α. An acceptable α is found by gradually reducing α to a value less than α_0 (which is signified by $f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k) > f(\boldsymbol{x}_k) + \rho\alpha \boldsymbol{g}_k^T \boldsymbol{d}_k$).

0.6 Newton Method

The steepest-descent method is a first-order method since it is based on the linear approximation of the Taylor series. It can be shown that if the line search in each GD iteration is performed exactly, the iterates generated by the GD algorithm form a zigzag trajectory as depicted in Fig. 0.3. As such, the GD algorithm can be slow, especially when the Hessian of the objective is *ill-conditioned* (i.e has a large *condition number*).

A second-order method known as the Newton (also known as the *Newton-Raphson*) method can be developed by using the quadratic approximation of the Taylor series:

$$f(\boldsymbol{x} + \boldsymbol{\delta}) \approx f(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})^T \boldsymbol{\delta} + \tfrac{1}{2}\boldsymbol{\delta}^T \boldsymbol{H}(\boldsymbol{x})\boldsymbol{\delta} \tag{0.3}$$

Let us consider the case where $\boldsymbol{H}(\boldsymbol{x})$ is positive definite. The right-hand side of (0.3) in this case is a *convex* quadratic approximation of the objective function in a vicinity of point $\boldsymbol{x}$, and this quadratic function has a unique minimizer which can be found by computing its gradient with respect to $\boldsymbol{\delta}$ and setting it to zero. This yields the optimum change in $\boldsymbol{x}$ as

$$\boldsymbol{\delta} = -\boldsymbol{H}(\boldsymbol{x})^{-1} \boldsymbol{g}(\boldsymbol{x}) \tag{0.4}$$

Now suppose a general *non-quadratic* function $f(\boldsymbol{x})$ is to be minimized and an arbitrary point $\boldsymbol{x}$ is assumed. Let us consider two scenarios:

(a) The Hessian $\boldsymbol{H}(\boldsymbol{x})$ is positive definite. The right-hand side of (0.3) in this case is a convex approximation of $f(\boldsymbol{x})$, hence the $\boldsymbol{\delta}$ produced by (0.4) is a reasonable change for updating point $\boldsymbol{x}$. This problem can be resolved by using an iterative procedure which incorporates a line search for the calculation of the change in $\boldsymbol{x}$. Namely this approach selects $\boldsymbol{x}_{k+1}$ as

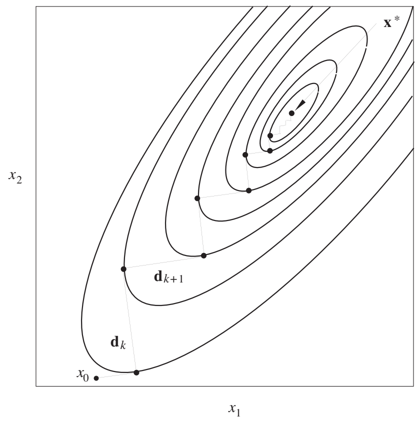


Fig. 0.3 Typical solution trajectory in the GD algorithm.

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k \tag{0.5a}$$

where

$$\boldsymbol{d}_k = -\boldsymbol{H}_k^{-1} \boldsymbol{g}_k \tag{0.5b}$$

and α_k is the value of α that minimizes $f(\boldsymbol{x}+\alpha\boldsymbol{d}_k)$. The vector $\boldsymbol{d}_k$ is referred to as the *Newton direction* at point $\boldsymbol{x}_k$.

(b) The Hessian $\boldsymbol{H}(\boldsymbol{x})$ is not positive definite. The $\boldsymbol{\delta}$ obtained from (0.3) in this case may not even yield a reduction in the objective function. The problem can be overcome by forcing $\boldsymbol{H}(\boldsymbol{x})$ to become positive definite by means of some manipulation (details are given below). Once the Hessian is modified to become positive definite, one is in scenario (a) again, which allows the iterative procedure to continue.

Figure 0.4 illustrates the first two Newton steps for the convex function

$$f(x) = x^2 + 0.05e^{-x} + 8$$

With $x_0 = -5$, the first two Newton iterations yield $x_1 = -3.15$ and $x_2 = -0.7929$ while the global minimizer is at $\boldsymbol{x}^* = 0.0242$.

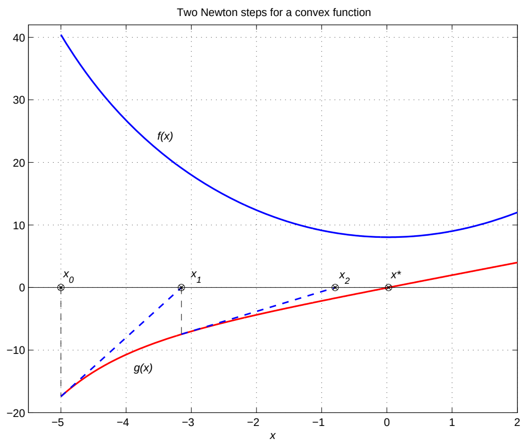


Figure 0.4 Two Newton steps for a convex function.

Newton algorithm

Step 1 Input $\boldsymbol{x}_0$ and initialize the tolerance ε. Set $k = 0$.

Step 2 Compute $\boldsymbol{g}_k$ and $\boldsymbol{H}_k$. If $\boldsymbol{H}_k$ is not positive definite, force it to become positive definite.

Step 3 Compute $\boldsymbol{H}_k^{-1}$ and $\boldsymbol{d}_k = -\boldsymbol{H}_k^{-1}\boldsymbol{g}_k$.

Step 4 Find α_k, the value of α that minimizes $f(\boldsymbol{x}+\alpha\boldsymbol{d}_k)$, using a line search.

Step 5 Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\boldsymbol{d}_k$. Compute $f_{k+1} = f(\boldsymbol{x}_{k+1})$.

Step 6 If $\|\alpha_k\boldsymbol{d}_k\| < \varepsilon$, then output $\boldsymbol{x}^* = \boldsymbol{x}_{k+1}$ and $f(\boldsymbol{x}^*) = f_{k+1}$, and stop; Otherwise, set $k = k + 1$ and repeat from Step 2.

Modification of the Hessian

If the Hessian is not positive definite in any iteration of the algorithm, it is forced to become positive definite in Step 2 of the algorithm. This modification of $\boldsymbol{H}_k$ can be accomplished in several ways. One of the approaches is to replace $\boldsymbol{H}_k$ by

$$\hat{\boldsymbol{H}}_k = \frac{\boldsymbol{H}_k + \beta\boldsymbol{I}_n}{1+\beta} \tag{0.6}$$

where $\boldsymbol{I}_n$ is the $n \times n$ identity matrix. If $\boldsymbol{H}_k$ is not positive definite, β in (0.6) is set to larger than the most negative eigenvalue of $\boldsymbol{H}_k$. If $\boldsymbol{H}_k$ is positive definite, β is set to zero so that the Hessian is not modified.

Note that when a large β is used,

$$\hat{\boldsymbol{H}}_k \approx \boldsymbol{I}_n$$

and from (0.5) the search direction becomes

$$\boldsymbol{d}_k \approx -\boldsymbol{g}_k$$

In effect, the modification in (0.6) converts the Newton method into the gradient descent method. A nonpositive definite $\boldsymbol{H}_k$ is likely to arise at points far from the solution where the gradient descent method is most effective in reducing the value of $f(\boldsymbol{x})$. Therefore, the modification in (0.6) leads to an algorithm that combines the complementary convergence properties of the Newton and gradient descent methods.

0.7 Quasi-Newton Methods

One may take a unifying look at the GD and Newton methods by writing their search directions as

$$\boldsymbol{d}_k = -\boldsymbol{S}_k\boldsymbol{g}_k \tag{0.7}$$

where

$$\boldsymbol{S}_k = \begin{cases} \boldsymbol{I} & \text{for gradient descent} \\ \boldsymbol{H}_k^{-1} & \text{for Newton} \end{cases}$$

Newton's method is known for its fast convergence and solution accuracy. However, this fast convergence is achieved at the cost of increased computational complexity due to the computation of the inverse Hessian in every Newton iteration. Quasi-Newton algorithms are developed to provide convergent rates comparable with that of Newton algorithm with reduced complexity based only on gradient information.

Quasi-Newton algorithms follow the general algorithmic structure described earlier, where search direction $\boldsymbol{d}_k$ is evaluated using (0.7) where matrix $\boldsymbol{S}_k$ is updated in each iteration using gradient information. The most well-known quasi-Newton algorithms is one developed by Broyden, Fletcher, Goldfarb, and Shanno (BFGS). The BFGS algorithm updates matrix $\boldsymbol{S}_k$ as follows.

◊ Quantities required: $\boldsymbol{\delta}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k, \quad \boldsymbol{\gamma}_k = \boldsymbol{g}_{k+1} - \boldsymbol{g}_k$

◊ Broyden-Fletcher-Goldfarb-Shanno (BFGS) updating formula: $\boldsymbol{S}_0 = \boldsymbol{I}$, and

$$\boldsymbol{S}_{k+1} = \boldsymbol{S}_k + \left(1 + \frac{\boldsymbol{\gamma}_k^T \boldsymbol{S}_k \boldsymbol{\gamma}_k}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k}\right) \frac{\boldsymbol{\delta}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k} - \frac{\boldsymbol{\delta}_k \boldsymbol{\gamma}_k^T \boldsymbol{S}_k + \boldsymbol{S}_k \boldsymbol{\gamma}_k \boldsymbol{\delta}_k^T}{\boldsymbol{\gamma}_k^T \boldsymbol{\delta}_k} \tag{0.8}$$

Example 0.2 Apply GD, Newton, and BFGS algorithms to the unconstrained minimization of the objective function

$$f(\boldsymbol{x}) = \sum_{i=1}^{199} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 \right]$$

Try the algorithms at two initial points:

$$\boldsymbol{x}_0^{(A)} = \texttt{[-ones(100,1); zeros(100,1)]}$$

$$\boldsymbol{x}_0^{(B)} = \texttt{[zeros(100,1); -ones(100,1)]}.$$

Solution: The objective function involves 200 variables. It is obvious that the global minimum is achieved at $\boldsymbol{x}^* =$ `ones(200,1)` at which $f(\boldsymbol{x}^*) = 0$.

To implement the algorithms, we evaluate the gradient and Hessian of $f(\boldsymbol{x})$ as follows:

• Gradient:

$$\frac{\partial f}{\partial x_1} = -400x_1\left(x_2 - x_1^2\right) + 2\left(x_1 - 1\right)$$

$$\frac{\partial f}{\partial x_i} = 200\left(x_i - x_{i-1}^2\right) - 400x_i\left(x_{i+1} - x_i^2\right) + 2\left(x_i - 1\right) \quad \text{for } 2 \le i \le 199$$

$$\frac{\partial f}{\partial x_{200}} = 200\left(x_{200} - x_{199}^2\right)$$

• Hessian:

$$\frac{\partial^2 f}{\partial x_1^2} = -400x_2 + 1200x_1^2 + 2$$

$$\frac{\partial^2 f}{\partial x_i \partial x_{i-1}} = \frac{\partial^2 f}{\partial x_{i-1} \partial x_i} = -400x_{i-1} \quad \text{for } 2 \le i \le 199$$

$$\frac{\partial^2 f}{\partial x_i^2} = 202 + 1200 x_i^2 - 400 x_{i+1} \quad \text{for } 2 \le i \le 199$$

$$\frac{\partial^2 f}{\partial x_i \partial x_{i+1}} = \frac{\partial^2 f}{\partial x_{i+1} \partial x_i} = -400 x_i \quad \text{for } 2 \le i \le 199$$

$$\frac{\partial^2 f}{\partial x_{200}^2} = 200$$

and the rest of the components of the Hessian are equal to zero.

All three algorithms converged to the global minimizer from the two initial points. However, performance in terms of algorithmic efficiency varied widely over these algorithms as one can see from the table below.

Algorithm	Initial Point	Tolerance	# of Iterations	$f(\boldsymbol{x}^*)$	Average CPU Time (Sec.)
GD	$\boldsymbol{x}_0^{(A)}$	10^{-9}	35763	4.9303×10^{-13}	34.6494
GD	$\boldsymbol{x}_0^{(B)}$	10^{-9}	21952	5.0286×10^{-13}	21.1693
Newton	$\boldsymbol{x}_0^{(A)}$	10^{-6}	312	0	5.5393
Newton	$\boldsymbol{x}_0^{(B)}$	10^{-6}	301	4.4775×10^{-30}	5.3757
BFGS	$\boldsymbol{x}_0^{(A)}$	10^{-7}	551	5.1087×10^{-15}	2.2761
BFGS	$\boldsymbol{x}_0^{(B)}$	10^{-7}	540	2.5731×10^{-14}	2.3486

From the table it is observed that GD is slow relative to the other algorithms. As expected, Newton algorithm took the smallest number of iterations to reach the solution with the best accuracy. However this does not mean Newton algorithm is the fastest because computing a Newton direction is expensive as it involves inverting a (possibly modified) Hessian of size 200 by 200. The complexity of Newton algorithm can be severe especially for problems whose sizes are not small enough. As a matter of fact, in the present case (with *n* = 200) Newton algorithm was clearly slower than BFGS. ■

Problems

0.1 With a real-valued constant κ, one can define a contour for a smooth objective function $f(\boldsymbol{x})$ as $\mathcal{C} = \{\boldsymbol{x} : f(\boldsymbol{x}) = \kappa\}$. For example, for a 2-variable function $f(x_1, x_2)$, $\mathcal{C}$ is a smooth curve on the x_1-x_2 space; for a 3-variable function $f(x_1, x_2, x_3)$, $\mathcal{C}$ is a smooth surface on the x_1-x_2-x_3 space. Let $\hat{\boldsymbol{x}}$ be a point on $\mathcal{C}$, $\nabla f(\hat{\boldsymbol{x}})$ be the gradient of $f(\boldsymbol{x})$ at $\hat{\boldsymbol{x}}$, and $\mathcal{T}$ be the tangent plane of contour $\mathcal{C}$ at $\hat{\boldsymbol{x}}$ (actually, for a 2-variable $f(x_1, x_2)$ $\mathcal{T}$ is a tangent line, while for a 3-variable $f(x_1, x_2, x_3)$ $\mathcal{T}$ is a tangent plane. For a general n-variable $f(\boldsymbol{x})$ we often call $\mathcal{T}$ a tangent plane). Show that $\nabla f(\hat{\boldsymbol{x}})$ is orthogonal to $\mathcal{T}$.

0.2 (a) Find all stationary points of the objective function

$$f(\boldsymbol{x}) = (x_1 + x_2)^2 + \left[2(x_1^2 + x_2^2 - 1) - \tfrac{1}{3}\right]^2$$

(b) Classify each stationary point found from part (a) as a minimizer, maximizer, or saddle point.

0.3 Apply GD, Newton, and BFGS algorithms to minimize the objective function given in Prob. 0.2 by doing the following:
(a) Generate a set of points that are evenly placed in the region $\mathcal{R}$ = $\{(x_1, x_2) : -2 \le x_1 \le 2, -2 \le x_2 \le 2\}$. The total number of point in this exercise is 43,264. This can be done using

```
x = -2:4/207:2; X0 = zeros(2,43264);
for i = 1:208, X0(:,(i-1)*208+1:i*208) = [x; x(209-i)*ones(1,208)];end
```

(b) Use each column of `X0` as an initial point and apply GD, Newton, and BFGS algorithms to minimize the objective function in Prob. 0.2. Set convergence tolerance $\varepsilon = 10^{-6}$, and in the case of Newton algorithm set `dt = 0.1`. Evaluate the value of the objective function at each solution point. As a result, you shall get three "maps", one from a specific algorithm, that displays the values of the objective function at the solution points that are obtained using initial points over the region $\mathcal{R}$. For each algorithm, use `mesh` to generate a 3-D plot showing the objective function values of the solution points when the initial points are from region $\mathcal{R}$.
(c) The 3-D plots produced from part (b) help visualize the performance of the algorithms applied. Comment on your results.

0.4 Solve the system of equations

$$f_1(\boldsymbol{x}) = x_1^3 - x_2 - 1 = 0$$
$$f_2(\boldsymbol{x}) = x_1^2 - x_2 = 0$$

by minimizing $F(\boldsymbol{x}) = f_1^2(\boldsymbol{x}) + f_2^2(\boldsymbol{x})$. Specifically, do the following:
(a) Verify that $\boldsymbol{x}^* = \begin{bmatrix}1.465571 & 2.147899\end{bmatrix}^T$ is a global minimizer of $F(\boldsymbol{x})$ that solves the above system of equations.
(b) Verify that $\hat{\boldsymbol{x}} = \begin{bmatrix}0 & -0.5\end{bmatrix}^T$ is a local minimizer of $F(\boldsymbol{x})$ that does not solve the system of equations.

(c) Verify that $\tilde{\boldsymbol{x}} = \begin{bmatrix} \frac{2}{3} & -\frac{7}{54} \end{bmatrix}^T$ is a saddle point of $F(\boldsymbol{x})$.

(d) Apply GD, Newton, and BFGS algorithms to minimize function $F(\boldsymbol{x})$ with initial point $\boldsymbol{x}_0 = \begin{bmatrix} 0.25 & 0.15 \end{bmatrix}^T$ which is considerably closer to the local minimizer $\hat{\boldsymbol{x}}$ than $\boldsymbol{x}^*$. Report and comment on the numerical results obtained.

0.5 Apply GD, Newton, and BFGS algorithms to minimize the objective function (known as the Beale function) given by

$$f(\boldsymbol{x}) = (x_1x_2 - x_1 + 1.5)^2 + (x_1x_2^2 - x_1 + 2.25)^2 + (x_1x_2^3 - x_1 + 2.625)^2$$

(a) Derive the gradient and Hessian of the Beale function.

(b) With each of the four initial points given below

$$\boldsymbol{x}_0^{(1)} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \quad \boldsymbol{x}_0^{(2)} = \begin{bmatrix} -3 \\ 3 \end{bmatrix}, \quad \boldsymbol{x}_0^{(3)} = \begin{bmatrix} 2 \\ -3 \end{bmatrix}, \quad \boldsymbol{x}_0^{(4)} = \begin{bmatrix} -3 \\ -3 \end{bmatrix}$$

and convergence tolerance $\varepsilon = 10^{-6}$, apply GD algorithm to minimize the Beale function. Report results in terms of the solution point found and the value of the objective function at the solution point with at least 8 decimal places and verify if the solution obtained is a local or global minimizer (explain why).

(c) With the same initial points and convergence tolerance as in part (b), apply Newton algorithm to minimize the Beale function. Report the solution point found and the value of the objective function at the solution point with at least 8 decimal places and verify if the solution obtained is a local or global minimizer (explain why).

(d) With the same initial points and convergence tolerance as in part (b), apply BFGS algorithm to minimize the Beale function. Report the solution point found and the value of the objective function at the solution point with at least 8 decimal places and verify if the solution obtained is a local or global minimizer (explain why).

(e) Compare and comment the results obtained in parts (b)-(d).

0.6 Repeat Example 0.2 with initial point $\boldsymbol{x}_0$ = `[-2*ones(100,1); 2*ones(100,1)]` and the following tolerance: $\varepsilon = 10^{-9}$ for GD; $\varepsilon = 10^{-6}$ for Newton; $\varepsilon = 10^{-8}$ for BFGS. Report numerical results with a table similar to that in Example 0.2 and comment the results obtained. MATLAB code for the four minimization algorithms involved as well as the backtracking-based line search algorithm can be found from the course website. To apply these algorithms to the problem at hand, however, you need to prepare function files for the objective function as well as its gradient and Hessian.

0.7 (a) Show that, for fixed vector $\boldsymbol{a}$ and positive scalar r,

$$\sup_{\boldsymbol{x}} \{\boldsymbol{a}^T\boldsymbol{x} : \|\boldsymbol{x}\|_2 \le r\} = r\|\boldsymbol{a}\|_2$$

where "$\sup_{\boldsymbol{x}}$" stands for *supremum* which by definition is the smallest upper bound of the set of numbers generated in $\{\cdot\}$ when $\boldsymbol{x}$ within the ball $\|\boldsymbol{x}\|_2 \le r$.

(b) Consider a symmetric matrix $\boldsymbol{X}$ which is partitioned into four blocks as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B} \\ \boldsymbol{B}^T & \boldsymbol{C} \end{bmatrix}$$

where $\boldsymbol{A}$ is assumed to be nonsingular. Show that

(i) there exists matrix $\boldsymbol{T}$ such that

$$\boldsymbol{T}^T \boldsymbol{X} \boldsymbol{T} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S} \end{bmatrix}$$

where $\boldsymbol{S} = \boldsymbol{C} - \boldsymbol{B}^T \boldsymbol{A}^{-1} \boldsymbol{B}$ is called the *Schur complement* of $\boldsymbol{A}$ in $\boldsymbol{X}$.

(ii) $\boldsymbol{X}$ is positive (semi)definite, if and only if both $\boldsymbol{A}$ and $\boldsymbol{S}$ are positive (semi)definite.

Chapter 1 Fundamentals of Constrained Optimization

1.1 Constrained Optimization Problems

In this course we study constrained optimization problems that can be formulated as

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) && \text{(1.1a)}\\ &\text{subject to:} && a_i(\boldsymbol{x}) = 0 \quad \text{for } i = 1,2,\ldots,p && \text{(1.1b)}\\ & && c_j(\boldsymbol{x}) \le 0 \quad \text{for } j = 1,2,\ldots,q && \text{(1.1c)} \end{aligned}$$

where $f(\boldsymbol{x})$ is *objective function*, $a_i(\boldsymbol{x}) = 0$ for i = 1, 2, …, p are *equality constraints*, $c_j(\boldsymbol{x}) \le 0$ for j = 1, 2, …, q are *inequality constraints*.

Collectively the constraints in (1.1b) and (1.1c) define a *feasible region*

$$\mathcal{R} = \{\boldsymbol{x} : a_i(\boldsymbol{x}) = 0, \text{ for } 1 \le i \le p \text{ and } c_j(\boldsymbol{x}) \le 0, \text{ for } 1 \le j \le q\}.$$

and $\boldsymbol{x}$ is said to be a *feasible point* if $\boldsymbol{x} \in \mathcal{R}$.

Unless stated otherwise, we assume that the objective function $f(\boldsymbol{x})$ as well as the constraint functions $a_i(\boldsymbol{x})$ for i = 1, 2, ..., p and $c_j(\boldsymbol{x})$ for j = 1, 2, ..., q, are continuous and twicely differentiable.

Example 1.1 Find the shortest distance between the two triangles shown in Fig. 1.1.

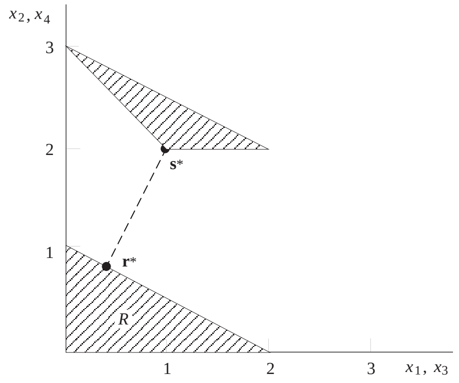


Figure 1.1

Solution Let $\boldsymbol{r} = [x_1 \quad x_2]^T \in R$ and $\boldsymbol{s} = [x_3 \quad x_4]^T \in S$. The squared distance between $\boldsymbol{r}$ and $\boldsymbol{s}$ is given by

$$(x_1 - x_3)^2 + (x_2 - x_4)^2 = \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x}$$

with

$$\boldsymbol{H} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

The two triangles in Fig. 1.1 are characterized by

$$R:\begin{cases} -x_1 \le 0 \\ -x_2 \le 0 \\ x_1 + 2x_2 - 2 \le 0 \end{cases} \qquad S:\begin{cases} -x_4 + 2 \le 0 \\ -x_3 - x_4 + 3 \le 0 \\ x_3 + 2x_4 - 6 \le 0 \end{cases}$$

Respectively, and these constraints can be expressed compactly as $\boldsymbol{Ax} \le \boldsymbol{b}$ where

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ -2 \\ -3 \\ 6 \end{bmatrix}$$

The above optimization problem can now be described as

$$\begin{array}{l} \text{minimize} \quad f(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{Hx} \\ \text{subject to:} \ \ \boldsymbol{Ax} \le \boldsymbol{b} \end{array} \tag{1.2}$$

which fits into the general formulation in (1.1).

1.2 Classes of Constrained Optimization Problems

Linear Programming (LP)

- **Standard-form LP:**

$$\begin{array}{l} \text{minimize} \quad \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to:} \ \ \boldsymbol{Ax} = \boldsymbol{b}, \ \boldsymbol{x} \ge \boldsymbol{0} \end{array} \tag{1.3}$$

- **Alternative-form LP:**

$$\begin{array}{l} \text{minimize} \quad \boldsymbol{c}^T \boldsymbol{x} \\ \text{subject to:} \ \ \boldsymbol{Ax} \le \boldsymbol{b} \end{array} \tag{1.4}$$

Quadratic Programming (QP)

- **QP**

$$\begin{array}{l} \text{minimize} \quad \tfrac{1}{2}\boldsymbol{x}^T \boldsymbol{Hx} + \boldsymbol{p}^T \boldsymbol{x} \\ \text{subject to:} \ \ \boldsymbol{Ax} \le \boldsymbol{b} \end{array} \tag{1.5}$$

where $\boldsymbol{H}$ is symmetric and positive semidefinite ($\boldsymbol{H} \succeq \boldsymbol{0}$).

- **Quadratically Constrained QP (QCQP)**

$$\begin{array}{l} \text{minimize} \quad \tfrac{1}{2}\boldsymbol{x}^T \boldsymbol{Hx} + \boldsymbol{p}^T \boldsymbol{x} \\ \text{subject to:} \ \ \boldsymbol{x}^T \boldsymbol{Q}_i \boldsymbol{x} + \boldsymbol{q}_i^T \boldsymbol{x} + r_i \le 0 \quad \text{for } 1 \le i \le q \end{array} \tag{1.6}$$

where $\boldsymbol{H}$ and $\boldsymbol{Q}_i$ are symmetric and positive semidefinite.

Semidefinite Programming (SDP)

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{c}^T \boldsymbol{x} \\ &\text{subject to: } \boldsymbol{F}(\boldsymbol{x}) \succeq \boldsymbol{0} \end{aligned} \tag{1.7}$$

where $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{F}_0 + \sum_{i=1}^{p} x_i \boldsymbol{F}_i$.

Second-Order Cone Programming (SOCP)

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{b}^T \boldsymbol{x} \\ &\text{subject to: } \left\| \boldsymbol{A}_j^T \boldsymbol{x} + \boldsymbol{c}_j \right\| \le \boldsymbol{b}_j^T \boldsymbol{x} + d_j \ \text{ for } 1 \le j \le q \end{aligned} \tag{1.8}$$

Convex Programming (CP)

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to: } \boldsymbol{a}_i^T \boldsymbol{x} = b_i \ \text{ for } 1 \le i \le p \\ &\qquad\qquad c_j(\boldsymbol{x}) \le 0 \ \text{ for } 1 \le j \le q \end{aligned} \tag{1.9}$$

where functions $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$ are convex. In other words, a CP problem minimizes a convex function over a convex feasible region.

Non-Differentiable Convex Problems

There are CP problems where either the objective function or some of the constrained functions or both are non-differentiable. It turns out that many constrained problems of importance belong to this problem class.

As an example, the l_1-minimization problem which plays an important role in *compressive sensing* is a nondifferentiable problem:

$$\begin{aligned} &\text{minimize} \quad \|\boldsymbol{x}\|_1 \\ &\text{subject to: } \boldsymbol{y} = \boldsymbol{\Phi} \boldsymbol{x} \end{aligned}$$

where $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$ is the l_1 norm of vector $\boldsymbol{x}$, $\boldsymbol{\Phi} \in R^{m \times n}$ with $m < n$ is a measurement matrix and $\boldsymbol{y} \in R^m$ collects m measurements. This is because the l_1 norm of $\boldsymbol{x}$ is non-differentiable with respect to $\boldsymbol{x}$.

General Nonconvex Constrained Problems

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to: } a_i(\boldsymbol{x}) = 0 \ \text{ for } i = 1, 2, \ldots, p \\ &\qquad\qquad c_j(\boldsymbol{x}) \ge 0 \ \text{ for } j = 1, 2, \ldots, q \end{aligned} \tag{1.10}$$

where either the objective function is nonconvex or the feasible region defined by the constraints is nonconvex, or both of them are nonconvex.

1.3 Lagrange Multipliers and Optimality Conditions

We begin with the notion of a point being regular.

Definition 1.1 Point $\boldsymbol{x}^*$ is said to be *regular* for a given set of equality constraints $\{a_i(\boldsymbol{x}) = 0, i = 1,$ $2, \ldots, p\}$ if the p vectors $\{\nabla a_i(\boldsymbol{x}^*), i = 1, 2...., p\}$ are linearly independent.

1.3.1 Optimality Conditions in Simple Cases

Definition 1.2 Let $\boldsymbol{x}$ be a feasible point for problem (1.1). Vector $\boldsymbol{d} \in R^n$ is said to be a *feasible direction* at $\boldsymbol{x}$ if there exists $\hat{\alpha} > 0$ such that $\boldsymbol{x} + \alpha \boldsymbol{d}$ remains feasible for all $\alpha \in [0, \hat{\alpha}]$.

Theorem 1.1 If $f(\boldsymbol{x}) \in C^1$ and $\boldsymbol{x}^*$ is a local minimizer of problem (1.1), then

$$\nabla f(\boldsymbol{x}^*)^T \boldsymbol{d} \geq 0 \quad \text{for every feasible direction } \boldsymbol{d} \text{ at } \boldsymbol{x}^* \tag{1.11}$$

Proof We examine Taylor's expansion of $f(\boldsymbol{x})$ at $\boldsymbol{x} = \boldsymbol{x}^* + \alpha \boldsymbol{d}$:

$$f(\boldsymbol{x}) = f(\boldsymbol{x}^*) + \alpha \nabla f(\boldsymbol{x}^*)^T \boldsymbol{d} + o(\alpha \| \boldsymbol{d} \|_2)$$

Hence

$$\alpha \nabla f(\boldsymbol{x}^*)^T \boldsymbol{d} + o(\alpha \| \boldsymbol{d} \|_2) = f(\boldsymbol{x}) - f(\boldsymbol{x}^*) \geq 0$$

which implies (1.11). ■

Example 1.2 Let $\boldsymbol{x}^*$ be a local minimizer of the equality constrained problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to:} \ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{1.12}$$

where $\boldsymbol{A} \in R^{p \times n}$ with $p < n$. The feasible region is defined by $\mathcal{R} = \{\boldsymbol{x}: \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}\}$. Obviously, $\boldsymbol{d}$ is a feasible direction at a feasible point $\boldsymbol{x}$ if and only if $\boldsymbol{A}\boldsymbol{d} = \boldsymbol{0}$, namely $\boldsymbol{d}$ belongs to the null space of $\boldsymbol{A}$. By applying Theorem 1.1, we obtain

$$\nabla f(\boldsymbol{x}^*)^T \boldsymbol{d} \geq 0 \quad \text{for } \boldsymbol{d} \in \mathcal{N}(\boldsymbol{A}) \tag{1.13}$$

where $\mathcal{N}(\boldsymbol{A})$ denotes the null space of $\boldsymbol{A}$. Since $\boldsymbol{d} \in \mathcal{N}(\boldsymbol{A})$ implies $-\boldsymbol{d} \in \mathcal{N}(\boldsymbol{A})$, (1.13) implies that $\nabla f(\boldsymbol{x}^*)^T \boldsymbol{d} = 0$ for $\boldsymbol{d} \in \mathcal{N}(\boldsymbol{A})$. From linear algebra, we conclude that

$$\nabla f(\boldsymbol{x}^*) \in \mathcal{N}^{\perp}(\boldsymbol{A}) = \mathcal{R}(\boldsymbol{A}^T) \tag{1.14}$$

This is to say that gradient $\nabla f(\boldsymbol{x}^*)$ is in the range of matrix $\boldsymbol{A}^T$. Therefore, there exists $\boldsymbol{\lambda} \in R^p$ (known as *Lagrange multiplier* for equality constraints) such that $\nabla f(\boldsymbol{x}^*) = -\boldsymbol{A}^T \boldsymbol{\lambda}$, namely,

$$\nabla f(\boldsymbol{x}^*) + \boldsymbol{A}^T \boldsymbol{\lambda} = \boldsymbol{0} \tag{1.15}$$

■

1.3.2 Equality Constraints

We now consider the constrained problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to: } a_i(\boldsymbol{x}) = 0 \ \text{ for } 1 \le i \le p \end{aligned} \tag{1.16}$$

Let $\boldsymbol{x}^*$ be a local minimizer of (1.16) and $\boldsymbol{s}$ be a feasible vector at $\boldsymbol{x}^*$. Thus we have $a_i(\boldsymbol{x}^*) = 0$ and $a_i(\boldsymbol{x}^* + \boldsymbol{s}) = 0$. The Taylor expansion of $a_i(\boldsymbol{x}^* + \boldsymbol{s})$ gives

$$0 = a_i(\boldsymbol{x}^* + \boldsymbol{s}) = a_i(\boldsymbol{x}^*) + \boldsymbol{s}^T \nabla a_i(\boldsymbol{x}^*) + o\left(\|\boldsymbol{s}\|\right) = \boldsymbol{s}^T \nabla a_i(\boldsymbol{x}^*) + o\left(\|\boldsymbol{s}\|\right)$$

which implies that

$$\boldsymbol{s}^T \nabla a_i(\boldsymbol{x}^*) = 0 \ \text{ for } 1 \le i \le p \tag{1.17}$$

In other words, $\boldsymbol{s}$ is feasible if and only if it is orthogonal to the gradients of the constraint functions. Now we project the gradient $\nabla f(\boldsymbol{x}^*)$ onto the space spanned by $\{\nabla a_1(\boldsymbol{x}^*), \nabla a_2(\boldsymbol{x}^*)$, $\ldots, \nabla a_p(\boldsymbol{x}^*)\}$. Let the projection be given by $-\sum_{i=1}^{p} \lambda_i^* \nabla a_i(\boldsymbol{x}^*)$, then $\nabla f(\boldsymbol{x}^*)$ can be expressed as

$$\nabla f(\boldsymbol{x}^*) = -\sum_{i=1}^{p} \lambda_i^* \nabla a_i(\boldsymbol{x}^*) + \boldsymbol{d} \tag{1.18}$$

where $\boldsymbol{d}$ is orthogonal to $\nabla a_i(\boldsymbol{x}^*)$ for i = 1, 2, …, p. Hence $\boldsymbol{s} = -\,\boldsymbol{d}$ is a feasible direction. From (1.17) and (1.18), it follows that $\boldsymbol{s}^T \nabla f(\boldsymbol{x}^*) = -\|\boldsymbol{d}\|^2$ which means that for $\boldsymbol{x}^*$ to be a local minimizer $\boldsymbol{d}$ must be zero (otherwise $\boldsymbol{s} = -\,\boldsymbol{d}$ would be a nontrivial feasible direction from $\boldsymbol{x}^*$ along which $f(\boldsymbol{x})$ would decrease, a contradiction with the assumption of $\boldsymbol{x}^*$ being a minimizer). This leads (1.18) to

$$\nabla f(\boldsymbol{x}^*) = -\sum_{i=1}^{p} \lambda_i^* \nabla a_i(\boldsymbol{x}^*) \tag{1.19}$$

The concept of Lagrange multiplier may also be understood from a different perspective. To see this we introduce the *Lagrangian* of problem (1.16) as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i a_i(\boldsymbol{x}) \tag{1.20}$$

where $\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 & \lambda_2 & \cdots & \lambda_p \end{bmatrix}^T$. In terms of the Lagrangian, (1.19) becomes

$$\nabla_x L(\boldsymbol{x},\boldsymbol{\lambda}) = \mathbf{0} \text{ for } \{\boldsymbol{x},\boldsymbol{\lambda}\} = \{\boldsymbol{x}^*,\boldsymbol{\lambda}^*\} \tag{1.21}$$

Eq. (1.21) offers n equations while the equality constraints in (1.16) contain p equations. Note that the number of equations available, $n + p$, matches the number of unknowns in $\boldsymbol{x}$ and $\boldsymbol{\lambda}$. Moreover, if we introduce the gradient operator ∇ as

$$\nabla = \begin{bmatrix} \nabla_x \\ \nabla_\lambda \end{bmatrix}$$

then these equations can be expressed as

$$\nabla L(\boldsymbol{x},\boldsymbol{\lambda}) = \mathbf{0} \text{ for } \{\boldsymbol{x},\boldsymbol{\lambda}\} = \{\boldsymbol{x}^*,\boldsymbol{\lambda}^*\} \tag{1.22}$$

In summary, the Lagrangian incorporates the constraints into a modified objective function in a way such that constrained minimizer $\boldsymbol{x}^*$ is connected to an unconstrained minimizer $\{\boldsymbol{x}^*,\boldsymbol{\lambda}^*\}$ for the augmented objective function $L(\boldsymbol{x},\boldsymbol{\lambda})$ where the augmentation is achieved with the p Lagrange multipliers.

Example 1.3 Solve the convex quadratic problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{p} \\ &\text{subject to: } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned}$$

where $\boldsymbol{H}$ is positive definite and $\boldsymbol{A} \in R^{p\times n}$ with $p \le n$ has full row rank.

Solution The Lagrangian of the problem is given by $L(\boldsymbol{x},\boldsymbol{\lambda}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{p} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$. Applying (1.22), we obtain

$$\nabla L(\boldsymbol{x},\boldsymbol{\lambda}) = \begin{bmatrix} \boldsymbol{H}\boldsymbol{x} + \boldsymbol{p} + \boldsymbol{A}^T\boldsymbol{\lambda} \\ \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H} & \boldsymbol{A}^T \\ \boldsymbol{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} + \begin{bmatrix} \boldsymbol{p} \\ -\boldsymbol{b} \end{bmatrix} = \mathbf{0}$$

Because $\boldsymbol{H} \succ \mathbf{0}$ and $\boldsymbol{A}$ has full row rank, one can show that matrix

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{A}^T \\ \boldsymbol{A} & \mathbf{0} \end{bmatrix}$$

is nonsingular (try to prove this fact as an exercise!). Therefore,

$$\begin{bmatrix} \boldsymbol{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{H} & \boldsymbol{A}^T \\ \boldsymbol{A} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} -\boldsymbol{p} \\ \boldsymbol{b} \end{bmatrix}$$

It follows that

$$\begin{aligned} \boldsymbol{x}^* &= -\boldsymbol{H}^{-1}(\boldsymbol{A}^T\boldsymbol{\lambda}^* + \boldsymbol{p}) \\ \boldsymbol{\lambda}^* &= -(\boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{A}^T)^{-1}(\boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{p} + \boldsymbol{b}) \end{aligned} \tag{1.23}$$

1.3.3 Inequality Constraints

First, a concept concerning inequality constraints. Suppose $\boldsymbol{x}^*$ is a minimizer of problem (1.1) and let $I(\boldsymbol{x}^*)$ be an index set $I(\boldsymbol{x}^*)$ = $\{j_1, j_2, \ldots, j_K\} \subseteq \{1, 2, \ldots, q\}$ for *active constraints* at $\boldsymbol{x}^*$, i.e., $c_j(\boldsymbol{x}^*) = 0$ for $j = j_1, j_2, \ldots, j_K$. As an example, Figure 1.2 shows a feasible region defined by R = $\{\boldsymbol{x} : c_j(\boldsymbol{x}) \le 0, \text{for } j = 1, 2, 3\}$ and a point $\bar{\boldsymbol{x}}$ at which constraint $c_3(\boldsymbol{x}) \le 0$ is active because $c_3(\bar{\boldsymbol{x}}) = 0$. We see that a point lies on the boundary of the feasible region if at the point some constraints become active.

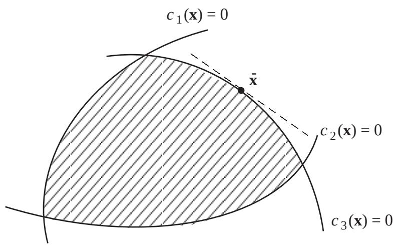


Figure 1.2

Now let us consider the general constrained problem (1.1):

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) && (1.1a)\\ &\text{subject to:} && a_i(\boldsymbol{x}) = 0 \quad \text{for } i = 1, 2, \ldots, p && (1.1b)\\ & && c_j(\boldsymbol{x}) \le 0 \quad \text{for } j = 1, 2, \ldots, q && (1.1c)\end{aligned}$$

and let $\boldsymbol{x}^*$ be a local minimizer of (1.1) at which there are K active inequality constraints, namely,

$$c_j(\boldsymbol{x}^*) = 0 \quad \text{for} \quad j \in I(\boldsymbol{x}^*) = \{j_1, j_2, \ldots, j_K\} \tag{1.24}$$

The K active constraints at $\boldsymbol{x}^*$ act like K additional equality constraints. Consequently, the equation in (1.19) need to be modified to

$$\nabla f(\boldsymbol{x}^*) = -\sum_{i=1}^{p} \lambda_i^* \nabla a_i(\boldsymbol{x}^*) - \sum_{k=1}^{K} \mu_{j_k}^* \nabla c_{j_k}(\boldsymbol{x}^*) \tag{1.25}$$

where $\mu_{j_k}^*$ are called Lagrange multipliers of the active inequality constraints.

An important property of the Lagrange multipliers associated with inequality constraints is that they are nonnegative, i.e.

$$\mu_{j_k}^* \ge 0 \text{ for } 1 \le k \le K \tag{1.26}$$

This property is illustrated in Fig. 1.3 for the case where the minimizer is in the interior of the feasible region (case (a)) and the case where the minimizer is on the boundary of the feasible region (case (b)).

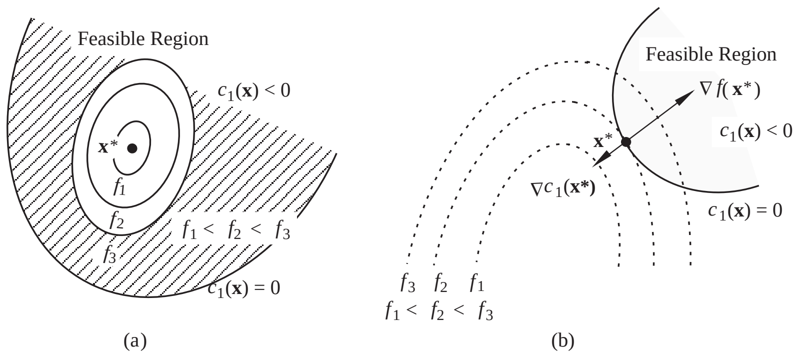


Figure 1.3

1.3.4 Karush-Kuhn-Tucker (KKT) Conditions

The KKT conditions are first-order necessary conditions that a local minimizer of problem (1.1) must satisfy.

Theorem 1.2 ***Karush-Kuhn-Tucker conditions*** *If* $\boldsymbol{x}^*$ *is a local minimizer of problem* (1.1) *and is regular for the constraints that are active at* $\boldsymbol{x}^*$*, then*

(a) $$a_i(\boldsymbol{x}^*) = 0 \text{ for } i = 1, 2, \ldots, p$$

(b) $$c_j(\boldsymbol{x}^*) \le 0 \text{ for } i = 1, 2, \ldots, q$$

(c) *There exist Lagrange multipliers* λ_i^* for $1 \le i \le p$ and μ_i^* for $1 \le i \le q$ *such that*

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{p} \lambda_i^* \nabla a_i(\boldsymbol{x}^*) + \sum_{j=1}^{q} \mu_j^* \nabla c_j(\boldsymbol{x}^*) = \mathbf{0} \tag{1.27}$$

(d) *Complementarity conditions*

$$\lambda_i^* a_i(\boldsymbol{x}^*) = 0 \text{ for } 1 \le i \le p \tag{1.28a}$$

$$\mu_j^* c_j(\boldsymbol{x}^*) = 0 \text{ for } 1 \le j \le q \tag{1.28b}$$

(e) $$\mu_j^* \ge 0 \text{ for } 1 \le j \le q \tag{1.29}$$

Definition 1.3 The Lagrangian for the general constrained problem (1.1) is defined by

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i a_i(\boldsymbol{x}) + \sum_{j=1}^{q} \mu_j c_j(\boldsymbol{x}) \tag{1.30}$$

Note that the condition in (1.27) can be expressed in terms of Lagrangian as

$$\nabla_x L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{0}$$

which is a set of n equations. These in combination with the p equations in condition (a) and q

equations in (1.28b) form a system of ($n + p + q$) equations for the same number of “unknowns” in $\boldsymbol{x}, \boldsymbol{\lambda}$ and $\boldsymbol{\mu}$. However, these equations in many cases are nonlinear and solving nonlinear system of equations is often no easier than solving the optimization problem itself. As a matter of fact, nonlinear equations are often solved by optimization techniques such as nonlinear least squares these days.

Example 1.4 Solve the constrained minimization problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = x_1^2 + x_2^2 - 14x_1 - 6x_2 \\ &\text{subject to:} \ c_1(\boldsymbol{x}) = -2 + x_1 + x_2 \le 0 \\ &\qquad\qquad\quad c_2(\boldsymbol{x}) = -3 + x_1 + 2x_2 \le 0 \end{aligned}$$

by applying the KKT conditions.

Solution The KKT conditions imply that

$$\begin{aligned} &2x_1 - 14 + \mu_1 + \mu_2 = 0 \\ &2x_2 - 6 + \mu_1 + 2\mu_2 = 0 \\ &\mu_1(-2 + x_1 + x_2) = 0 \\ &\mu_2(-3 + x_1 + 2x_2) = 0 \\ &\mu_1 \ge 0 \\ &\mu_2 \ge 0 \end{aligned}$$

One way to find the solution in this simple case is to consider all possible cases with regard to active constraints and verify the nonnegativity of the $\mu_j's$ obtained.

Case 1: *No active constraints*. If there are no active constraints, we have $\mu_1^* = \mu_2^* = 0$, which leads to

$$\boldsymbol{x}^* = \begin{bmatrix} 7 \\ 3 \end{bmatrix}$$

Obviously, this $\boldsymbol{x}^*$ violates both constraints and it is not a solution.

Case 2: *One constraint is active.* If only the first constraint is active, then we have $\mu_2^* = 0$, and

$$\begin{aligned} &2x_1 - 14 + \mu_1 = 0 \\ &2x_2 - 6 + \mu_1 = 0 \\ &-2 + x_1 + x_2 = 0 \end{aligned}$$

Solving this system of equations, we obtain

$$\boldsymbol{x}^* = \begin{bmatrix} 3 \\ -1 \end{bmatrix} \quad \text{and} \quad \mu_1^* = 8$$

Since $\boldsymbol{x}^*$ also satisfies the second constraint, $\boldsymbol{x}^* = [3 \quad -1]^T$ and $\boldsymbol{\mu}^* = [8 \quad 0]^T$ satisfy the KKT conditions.

If only the second constraint is active, then $\mu_1^* = 0$ and the KKT conditions become

$$\begin{aligned}2x_1 - 14 + \mu_2 &= 0\\ 2x_2 - 6 + 2\mu_2 &= 0\\ -3 + x_1 + 2x_2 &= 0\end{aligned}$$

The solution of this system of equations is given by

$$\boldsymbol{x}^* = \begin{bmatrix}5\\-1\end{bmatrix} \text{ and } \mu_2^* = 4$$

As $\boldsymbol{x}^*$ violates the first constraint, the above $\boldsymbol{x}^*$ and $\boldsymbol{\mu}^*$ do not satisfy the KKT conditions.

Case 3: *Both constraints are active*. If both constraints are active, we have

$$\begin{aligned}2x_1 - 14 + \mu_1 + \mu_2 &= 0\\ 2x_2 - 6 + \mu_1 + 2\mu_2 &= 0\\ -2 + x_1 + x_2 &= 0\\ -3 + x_1 + 2x_2 &= 0\end{aligned}$$

The solution to this system of equations is given by

$$\boldsymbol{x}^* = \begin{bmatrix}1\\1\end{bmatrix} \text{ and } \boldsymbol{\mu}^* = \begin{bmatrix}20\\-8\end{bmatrix}$$

Since $\mu_2^* < 0$, this is not a solution of the optimization problem.

Therefore, the only candidate for a minimizer of the problem is given by

$$\boldsymbol{x}^* = \begin{bmatrix}3\\-1\end{bmatrix} \text{ and } \boldsymbol{\mu}^* = \begin{bmatrix}8\\0\end{bmatrix}$$

As can be observed in Fig. 1.4, the above point is actually the global minimizer.

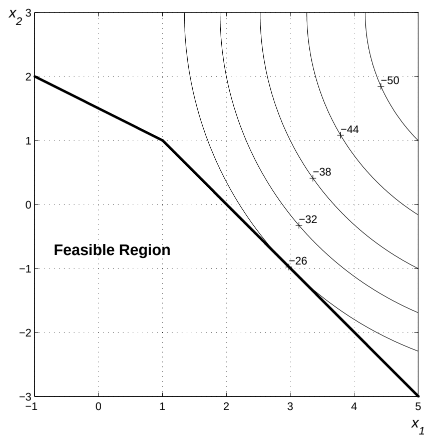


Figure 1.4

1.4 Second-Order Conditions

1.4.1 Second-Order Necessary Conditions

Theorem 1.3 (***Second-order necessary conditions for a minimum, general constrained problem***) *If* $\boldsymbol{x}^*$ *is a constrained local minimizer of problem (1.1) and is a regular point of the constraints in (1.1b) and (1.1c), then*

(*a*) $$a_i(\boldsymbol{x}^*) = 0 \text{ for } i = 1, 2, \ldots, p$$

(*b*) $$c_j(\boldsymbol{x}^*) \le 0 \text{ for } i = 1, 2, \ldots, q$$

(*c*) *there exist Lagrange multipliers* λ_i^* for $1 \le i \le p$ and μ_i^* for $1 \le i \le q$ *such that*

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{p} \lambda_i^* \nabla a_i(\boldsymbol{x}^*) + \sum_{j=1}^{q} \mu_j^* \nabla c_j(\boldsymbol{x}^*) = \mathbf{0}$$

(*d*) *Complementarity conditions*

$$\lambda_i^* a_i(\boldsymbol{x}^*) = 0 \text{ for } 1 \le i \le p$$

$$\mu_j^* c_j(\boldsymbol{x}^*) = 0 \text{ for } 1 \le j \le q$$

(*e*) $$\mu_j^* \ge 0 \text{ for } 1 \le j \le q$$

(*f*) $$\boldsymbol{N}^T(\boldsymbol{x}^*) \nabla_x^2 L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \boldsymbol{N}(\boldsymbol{x}^*) \succeq \mathbf{0} \tag{1.31}$$

Matrix $\boldsymbol{N}(\boldsymbol{x}^*)$ in (1.31) is composed of basis vectors of the null space of Jacobian $\boldsymbol{J}(\boldsymbol{x}^*)$ which is defined by

$$\boldsymbol{J}(\boldsymbol{x}^*) = \begin{bmatrix} \boldsymbol{J}_e(\boldsymbol{x}^*) \\ \boldsymbol{J}_{ie}(\boldsymbol{x}^*) \end{bmatrix} \tag{1.32}$$

with $\boldsymbol{J}_e(\boldsymbol{x}^*)$ the Jacobian of the equality constraint functions, i.e.

$$\boldsymbol{J}_e(\boldsymbol{x}^*) = \begin{bmatrix} \nabla a_1(\boldsymbol{x}^*) & \nabla a_2(\boldsymbol{x}^*) & \cdots & \nabla a_p(\boldsymbol{x}^*) \end{bmatrix}^T \tag{1.33}$$

and $\boldsymbol{J}_{ie}(\boldsymbol{x}^*)$ the Jacobian of the inequality constraint functions that are active at $\boldsymbol{x}^*$, i.e.

$$\boldsymbol{J}_{ie}(\boldsymbol{x}^*) = \begin{bmatrix} \nabla c_{j_1}(\boldsymbol{x}^*) & \nabla c_{j_2}(\boldsymbol{x}^*) & \cdots & \nabla c_{j_K}(\boldsymbol{x}^*) \end{bmatrix}^T \tag{1.34}$$

To understand condition (1.31), recall the Lagrangian for problem (1.1) which is defined by

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i a_i(\boldsymbol{x}) + \sum_{j=1}^{q} \mu_j c_j(\boldsymbol{x}) \tag{1.30}$$

and let $\boldsymbol{x}^*$ be a local minimizer of (1.1) and $\{\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*\}$ satisfies the KKT conditions. Note that

$$L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) = f(\boldsymbol{x}^*) \tag{1.35}$$

A direction $\boldsymbol{s}$ is said to be feasible if

$$\begin{aligned} a_i(\boldsymbol{x}^* + \boldsymbol{s}) &= 0 \quad \text{for } 1 \le i \le p \\ c_j(\boldsymbol{x}^* + \boldsymbol{s}) &= 0 \quad \text{for } j \in I(\boldsymbol{x}^*) \end{aligned} \tag{1.36}$$

From (1.27), (1.30), (1.35), and (1.36), we have

$$\begin{aligned} f(\boldsymbol{x}^* + \boldsymbol{s}) &= L(\boldsymbol{x}^* + \boldsymbol{s}, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \\ &= L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) + \boldsymbol{s}^T \nabla_x L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) + \tfrac{1}{2} \boldsymbol{s}^T \nabla_x^2 L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \boldsymbol{s} + o(\|\boldsymbol{s}\|^2) \\ &= f(\boldsymbol{x}^*) + \tfrac{1}{2} \boldsymbol{s}^T \nabla_x^2 L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \boldsymbol{s} + o(\|\boldsymbol{s}\|^2) \end{aligned}$$

which in conjunction with that fact that $f(\boldsymbol{x}^*) \le f(\boldsymbol{x}^* + \boldsymbol{s})$ implies that

$$\boldsymbol{s}^T \nabla_x^2 L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \boldsymbol{s} \ge 0 \tag{1.37}$$

for any $\boldsymbol{s}$ feasible at $\boldsymbol{x}^*$. To characterize all feasible $\boldsymbol{s}$, we use (1.36) to write

$$\begin{aligned} 0 &= a_i(\boldsymbol{x}^* + \boldsymbol{s}) = a_i(\boldsymbol{x}^*) + \nabla^T a_i(\boldsymbol{x}^*) \boldsymbol{s} + O(\|\boldsymbol{s}\|^2) = \nabla^T a_i(\boldsymbol{x}^*) \boldsymbol{s} + O(\|\boldsymbol{s}\|^2) \Rightarrow \nabla^T a_i(\boldsymbol{x}^*) \boldsymbol{s} = 0 \\ 0 &= c_{j_k}(\boldsymbol{x}^* + \boldsymbol{s}) = c_{j_k}(\boldsymbol{x}^*) + \nabla^T c_{j_k}(\boldsymbol{x}^*) \boldsymbol{s} + O(\|\boldsymbol{s}\|^2) = \nabla^T c_{j_k}(\boldsymbol{x}^*) \boldsymbol{s} + O(\|\boldsymbol{s}\|^2) \Rightarrow \nabla^T c_{j_k}(\boldsymbol{x}^*) \boldsymbol{s} = 0 \end{aligned}$$

Hence

$$\boldsymbol{J}(\boldsymbol{x}^*) \boldsymbol{s} = \begin{bmatrix} \boldsymbol{J}_e(\boldsymbol{x}^*) \\ \boldsymbol{J}_{ie}(\boldsymbol{x}^*) \end{bmatrix} \boldsymbol{s} = \boldsymbol{0} \tag{1.38}$$

This is to say that all feasible $\boldsymbol{s}$ lie in the null space of $\boldsymbol{J}(\boldsymbol{x}^*)$ and shows the equivalence of condition (1.37) and (1.31).

• We remark that for a constrained problem with only equality (or only inequality) constraints, the matrix $\boldsymbol{N}(\boldsymbol{x}^*)$ in (1.31) is composed of basis vectors of the null space of Jacobian $\boldsymbol{J}_e(\boldsymbol{x}^*)$ (or $\boldsymbol{J}_{ie}(\boldsymbol{x}^*)$).

Example 1.5 (i) Find points that satisfy the KKT conditions for the constrained problem

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) = x_1^2 + x_2^2 + \tfrac{1}{4} x_3^2 \\ \text{subject to:} \quad & a_1(\boldsymbol{x}) = -x_1 + x_3 - 1 = 0 \\ & a_2(\boldsymbol{x}) = x_1^2 + x_2^2 - 2x_1 = 0 \end{aligned}$$

(ii) Check the 2nd-order necessary conditions for the points found in part (i).

Solution (i) We compute

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} 2x_1 \\ 2x_2 \\ \tfrac{1}{2} x_3 \end{bmatrix}, \quad J_e^T(\boldsymbol{x}) = \begin{bmatrix} \nabla a_1(\boldsymbol{x}) & \nabla a_2(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} -1 & 2x_1 - 2 \\ 0 & 2x_2 \\ 1 & 0 \end{bmatrix}$$

Hence (1.27) becomes

$$\begin{aligned} & 2x_1 - \lambda_1 + \lambda_2(2x_1 - 2) = 0 \\ & 2x_2 + 2\lambda_2 x_2 = 0 \\ & x_3 + 2\lambda_1 = 0 \end{aligned}$$

$$-x_1 + x_3 - 1 = 0$$
$$x_1^2 + x_2^2 - 2x_1 = 0$$

Solving the above system of equations, we obtain two solutions:

$$\boldsymbol{x}_1^* = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \text{ and } \boldsymbol{\lambda}_1^* = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{4} \end{bmatrix}$$

and

$$\boldsymbol{x}_2^* = \begin{bmatrix} 2 \\ 0 \\ 3 \end{bmatrix} \text{ and } \boldsymbol{\lambda}_2^* = \begin{bmatrix} -\frac{3}{2} \\ -\frac{11}{4} \end{bmatrix}$$

(ii) To check the second-order necessary conditions, we need to evaluate the Largangian and Jacobian at the points found above.

At $\boldsymbol{x}_1^* = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ and $\boldsymbol{\lambda}_1^* = \begin{bmatrix} -\frac{1}{2} & \frac{1}{4} \end{bmatrix}^T$, we compute

$$\nabla_x^2 L(\boldsymbol{x}_1^*, \boldsymbol{\lambda}_1^*) = \begin{bmatrix} 2.5 & 0 & 0 \\ 0 & 2.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$

Since the Hessian of the Lagrangian is positive definite regardless of the null space of $\boldsymbol{J}_e(\boldsymbol{x}_1^*)$, the 2nd-order necessary conditions are satisfied.

At $\boldsymbol{x}_2^* = \begin{bmatrix} 2 & 0 & 3 \end{bmatrix}^T$ and $\boldsymbol{\lambda}_2^* = \begin{bmatrix} -\frac{3}{2} & -\frac{11}{4} \end{bmatrix}^T$, the Hessian of the Lagrangian becomes

$$\nabla^2 L(\boldsymbol{x}_2^*, \boldsymbol{\lambda}_2^*) = \begin{bmatrix} -3.5 & 0 & 0 \\ 0 & -3.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix}$$

The Jacobian $\boldsymbol{J}_e(\boldsymbol{x}_2^*)$ is found to be

$$\boldsymbol{J}_e(\boldsymbol{x}_2^*) = \begin{bmatrix} -1 & 0 & 1 \\ 2 & 0 & 0 \end{bmatrix}$$

whose null space is spanned by $\boldsymbol{N}(\boldsymbol{x}_2^*) = [0 \quad 1 \quad 0]^T$. Since

$$\boldsymbol{N}^T(\boldsymbol{x}_2^*) \nabla_x^2 L(\boldsymbol{x}_2^*, \boldsymbol{\lambda}_2^*) \boldsymbol{N}(\boldsymbol{x}_2^*) = -3.5 < 0$$

$\{\boldsymbol{x}_2^*, \boldsymbol{\lambda}_2^*\}$ does not satisfy the 2nd-order necessary conditions. ■

1.4.2 Second-Order Sufficient Conditions

Theorem 1.4 ***Second-order sufficient conditions for a minimum, general constrained problem***
A point $\boldsymbol{x}^*$ is a strong local minimizer of problem (1.1) if

(a) $$a_i(\boldsymbol{x}^*) = 0 \text{ for } i = 1,2,\ldots,p$$

(b) $$c_j(\boldsymbol{x}^*) \le 0 \text{ for } i = 1,2,\ldots,q$$

(c) $\boldsymbol{x}^*$ *is a regular point of the constraints that are active at* $\boldsymbol{x}^*$.

(d) *there exist Lagrange multipliers* λ_i^* for $1 \le i \le p$ and μ_i^* for $1 \le i \le q$ *such that*

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{p} \lambda_i^* \nabla a_i(\boldsymbol{x}^*) + \sum_{j=1}^{q} \mu_j^* \nabla c_j(\boldsymbol{x}^*) = \mathbf{0}$$

(e) *Complementarity conditions*

$$\lambda_i^* a_i(\boldsymbol{x}^*) = 0 \text{ for } 1 \le i \le p$$

$$\mu_j^* c_j(\boldsymbol{x}^*) = 0 \text{ for } 1 \le j \le q$$

(f) $$\mu_j^* \ge 0 \text{ for } 1 \le j \le q$$

(g) $$\boldsymbol{N}^T(\boldsymbol{x}^*)\nabla_x^2 L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)\boldsymbol{N}(\boldsymbol{x}^*) \succ \mathbf{0} \quad (1.39)$$

where $\boldsymbol{N}(\boldsymbol{x}^*)$ whose columns form a basis of the null space of Jacobian $\tilde{\boldsymbol{J}}(\boldsymbol{x}^*)$ defined by

$$\tilde{\boldsymbol{J}}(\boldsymbol{x}^*) = \begin{bmatrix} \boldsymbol{J}_e(\boldsymbol{x}^*) \\ \tilde{\boldsymbol{J}}_{ie}(\boldsymbol{x}^*) \end{bmatrix} \quad (1.40)$$

in that the Jacobian $\tilde{\boldsymbol{J}}_{ie}(\boldsymbol{x}^*)$ is the matrix whose rows are composed of those gradients of inequality constraints that are active at $\boldsymbol{x}^*$ and $\mu_j^* > 0$.

As an example, one can apply the 2nd-order sufficient conditions to the problem in Example 1.5 to conclude that the candidate for a local minimum, namely $\boldsymbol{x}^* = [3\ -1]^T$ and $\boldsymbol{\mu}^* = [8 \quad 0]^T$ is a strong local minimizer because

$$\nabla^2 L(\boldsymbol{x}^*, \boldsymbol{\mu}^*) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \succ \mathbf{0}.$$

·As another example, one can apply the 2nd-order sufficient conditions to the problem in Example 1.5 to conclude that $\boldsymbol{x}_1^* = [0 \quad 0 \quad 1]^T$ is a strong local minimizer because

$$\nabla^2 L(\boldsymbol{x}_1^*, \boldsymbol{\lambda}_1^*) = \begin{bmatrix} 2.5 & 0 & 0 \\ 0 & 2.5 & 0 \\ 0 & 0 & 0.5 \end{bmatrix} \succ \mathbf{0}$$

Problems

1.1 A trigonometric polynomial is given by

$$A(\omega) = \sum_{k=0}^{n} a_k \cos k\omega \tag{P1.1}$$

and Ω_p, Ω_a are sets given by

$$\Omega_p = \{\omega_{p0}, \omega_{p1}, \cdots, \omega_{pN}\} \subseteq [0, \omega_p]$$
$$\Omega_a = \{\omega_{a0}, \omega_{a1}, \cdots, \omega_{aM}\} \subseteq [\omega_a, \pi]$$

with $\omega_p \le \omega_a$. Coefficients a_k for k = 0, 1, …, n are required in (P1.1) such that the upper bound δ in

$$|A(\omega) - 1| \le \delta \quad \text{for } \omega \in \Omega_p \tag{P1.2}$$

and

$$|A(\omega)| \le \delta \quad \text{for } \omega \in \Omega_a \tag{P1.3}$$

is minimized. Formulate the above problem as a constrained minimization problem as seen in (1.1). Note that in the formulation the upper bound δ should be treated as an additional design variable.

1.2 Consider the trigonometric polynomial $A(\omega)$ given in Prob.1.1. Suppose we need to find a_k for k = 0, 1, …, n such that

$$J = \int_0^{\omega_p} [A(\omega) - 1]^2 d\omega + \int_{\omega_a}^{\pi} W(\omega) A(\omega)^2 d\omega \tag{P1.4}$$

is minimized subject to the constraints in (P1.2) and (P1.3), where $W(\omega) \ge 0$ is a weighting function and δ is treated as a known positive scalar. Formulate the above problem as a constrained minimization problem as seen in Eq. (1.1).

1.3 Consider the LP problem

$$\begin{array}{ll} \text{mimimize} & c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{subject to:} & a_i \le x_i \le b_i \quad \text{for } i = 1, 2, \ldots, n \end{array}$$

where a_i, b_i and c_i are constants and all c_i are nonzero. Show that the solution of the LP problem is given by $\boldsymbol{x}^* = \begin{bmatrix} x_1^* & x_2^* & \cdots & x_n^* \end{bmatrix}^T$ with

$$x_i^* = \begin{cases} a_i & \text{if } c_i > 0 \\ b_i & \text{if } c_i < 0 \end{cases}$$

1.4 Consider the constrained problem

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to:} & \boldsymbol{x} \ge \boldsymbol{0} \end{array}$$

where $f(\boldsymbol{x}) \in C^1$. If $\boldsymbol{x}^*$ is a local minimizer of the problem, then the following hold true:

$$\begin{cases} \text{if } x_i^* = 0, \text{ then } \left(\nabla f(\boldsymbol{x}^*)\right)_i \geq 0 \\ \text{if } x_i^* > 0, \text{ then } \left(\nabla f(\boldsymbol{x}^*)\right)_i = 0 \end{cases}$$

1.5 Consider the constrained minimization problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to:} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\qquad\qquad c_j(\boldsymbol{x}) \leq 0 \ \text{ for } j = 1, 2, \ldots, q \end{aligned} \tag{P1.5a-c}$$

Show that problem (P1.5) is equivalent to the problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) + \sum_{j=1}^{q} I_-(c_j(\boldsymbol{x})) \\ &\text{subject to:} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{P1.6a-b}$$

where $I_-(u)$ denotes the *indicator function* for nonpositive real *u*:

$$I_-(u) = \begin{cases} 0 & \text{if } u \leq 0 \\ \infty & \text{if } u > 0 \end{cases} \tag{P1.7}$$

1.6 Evaluate and compare function $h(u) = -(1/\tau)\log(-u)$ over $u < 0$ with the indicator function $I_-(u)$ defined in (P1.7), where $\tau > 0$ is a parameter that controls the accuracy of the approximation:

(a) Plot the indicator function $I_-(u)$ over $u < 0$.

(b) In the same plot, draw function $h(u)$ over the interval [–2, 0) for $\tau = 0.5$, 1, and 2. What can you conclude from the figures?

(c) Based on part (b), formulate a "more friendly" (i.e., differentiable) optimization problem that approximates problem (P1.6) whose objective function is non-differentiable.

1.7 Consider the minimization problem involving a scalar variable *x*:

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{subject to:} \quad a \leq x \leq b \end{aligned} \tag{P1.8}$$

where the objective $f(x)$ is a smooth function that is monotonically increasing over the feasible region [*a*, *b*], see the figure below for illustration. Applying the KKT conditions to show that the minimum of the objective is achieved at *x* = *a*. Hint: Show that at any point in the feasible interval [*a*, *b*] other than *x* = *a*, at least one of the KKT conditions is violated.

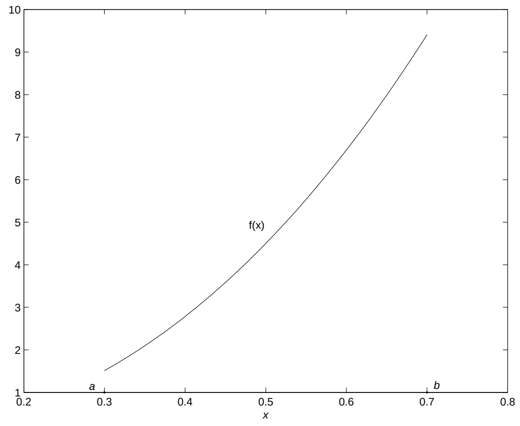


Figure for Problem 1.7

1.8 Consider the minimization problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = x_1^2 + (x_2 - 2)^2 \\ &\text{subject to:} \ \ a(\boldsymbol{x}) = \gamma x_1^2 - x_2 = 0 \end{aligned} \tag{P1.9}$$

where γ is a constant parameter.

(i) Give the KKT conditions of the problem in (P1.9) explicitly. Does point $\boldsymbol{x}^* = [0 \quad 0]^T$ satisfy these conditions?

(ii) Is $\boldsymbol{x}^* = [0 \quad 0]^T$ a minimizer of the problem and why? (Hint: the answer has a lot to do with the numerical value of γ. You don't have to apply second-order conditions to resolve the problem, just have a careful examination of the objective function at feasible points inside a small vicinity surrounding point $\boldsymbol{x}^* = [0 \quad 0]^T$.

1.9 Use the Lagrange multiplier method to solve the QP problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{p}^T\boldsymbol{x} \\ &\text{subject to:} \qquad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned}$$

where

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_1 & \boldsymbol{H}_2 & \boldsymbol{H}_3 & \boldsymbol{H}_4 \\ \boldsymbol{H}_2 & \boldsymbol{H}_1 & \boldsymbol{H}_2 & \boldsymbol{H}_3 \\ \boldsymbol{H}_3 & \boldsymbol{H}_2 & \boldsymbol{H}_1 & \boldsymbol{H}_2 \\ \boldsymbol{H}_4 & \boldsymbol{H}_3 & \boldsymbol{H}_2 & \boldsymbol{H}_1 \end{bmatrix}$$

with

$$\boldsymbol{H}_1 = \begin{bmatrix} 14 & 8 & 7 & 6 \\ 8 & 14 & 8 & 7 \\ 7 & 8 & 14 & 8 \\ 6 & 7 & 8 & 14 \end{bmatrix}, \quad \boldsymbol{H}_2 = \begin{bmatrix} 3 & 2 & 1 & 0 \\ 2 & 3 & 2 & 1 \\ 1 & 2 & 3 & 2 \\ 0 & 1 & 2 & 3 \end{bmatrix}, \quad \boldsymbol{H}_3 = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \quad \boldsymbol{H}_4 = \boldsymbol{I}_4$$

$$p = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}, \quad p_1 = \begin{bmatrix} 1 \\ -1 \\ 2 \\ -2 \end{bmatrix}, \quad p_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad p_3 = \begin{bmatrix} 2 \\ 2 \\ -4 \\ 4 \end{bmatrix}, \quad p_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\boldsymbol{A} = [\boldsymbol{H}_3 \quad \boldsymbol{H}_2 \quad \boldsymbol{H}_1 \quad \boldsymbol{H}_4], \quad \boldsymbol{b} = [1 \quad 8 \quad 3 \quad -4]^T.$$

1.10 Consider the constrained minimization problem

$$\begin{aligned} &\text{minimize } f(\boldsymbol{x}) = -2x_1^2 x_2 - x_2^3 + x_3^3 \\ &\text{subject to: } x_1 + 2x_2 + x_3^2 - 5 = 0 \\ &\qquad\qquad x_1 - 5x_2^2 + x_3^2 \le -2 \end{aligned}$$

$$x_1 \ge 0, x_2 \ge 0, x_3 \ge 0$$

(a) Write the KKT conditions for the solution points of the problem

(b) Vector $\boldsymbol{x}^* = [3 \ 1 \ 0]^T$ is known to be a local minimizer. At $\boldsymbol{x}^*$, find λ_1^* and μ_i^* for $1 \le i \le 4$, and

verify that $\mu_i^* \ge 0$ for $1 \le i \le 4$.

(c) Examine the second-order conditions for set $\{\boldsymbol{x}^*, \lambda^*, \boldsymbol{\mu}^*\}$.

1.11 Consider the minimization problem

$$\begin{aligned} &\text{minimize } f(\boldsymbol{x}) = \boldsymbol{c}^T \boldsymbol{x} \\ &\text{subject to: } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0} \\ &\qquad\qquad \|\boldsymbol{x}\|_2 \le 1 \end{aligned}$$

(a) Show that this is a CP problem.
(b) Derive the KKT conditions for the solution points of the problem.

(c) Show that if $\boldsymbol{c} - \boldsymbol{A}^T \boldsymbol{\lambda} \ne \boldsymbol{0}$ where $\boldsymbol{\lambda}$ satisfies

$$\boldsymbol{A}\boldsymbol{A}^T \boldsymbol{\lambda} = \boldsymbol{A}\boldsymbol{c}$$

then the minimizer is given by

$$\boldsymbol{x}^* = -\frac{\boldsymbol{c} - \boldsymbol{A}^T \boldsymbol{\lambda}}{\|\boldsymbol{c} - \boldsymbol{A}^T \boldsymbol{\lambda}\|_2}$$

Otherwise, any feasible $\boldsymbol{x}$ is a solution.

1.12 Consider the minimization problem

$$\begin{aligned} &\text{minimize } f(\boldsymbol{x}) = \boldsymbol{c}^T \boldsymbol{x} \\ &\text{subject to: } \|\boldsymbol{A}\boldsymbol{x}\|_2 \le 1 \end{aligned}$$

(a) Show that this is a CP problem.
(b) Derive the KKT conditions for the solution points of the problem.
(c) Show that if the solution of the equation

$$\boldsymbol{A}^T \boldsymbol{A}\boldsymbol{y} = \boldsymbol{c}$$

is nonzero, then the minimizer is given by

$$\boldsymbol{x}^* = -\frac{\boldsymbol{y}}{\| \boldsymbol{A}\boldsymbol{y} \|_2}$$

Otherwise, any feasible $\boldsymbol{x}$ is a solution.

1.13 Consider the minimization problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}^T \boldsymbol{x} \\ &\text{subject to:} \quad \boldsymbol{x}^T \boldsymbol{x} \le 1 \end{aligned}$$

where $\boldsymbol{c} \ne \boldsymbol{0}$.

(a) Drive a closed-form solution of the problem via its KKT conditions.

(b) Drive a closed-form solution of the problem via the result of Prob. 1.12.

Chapter 2 Convexity and Duality

A distinguishing idea which dominates many issues in optimization theory is convexity. Before we proceed, we quote the following from R. T. Rockafellar "Lagrange multipliers and optimality", *SIAM Review*, vol. 35, no. 2, pp. 183-238, June 1993: "Convexity is a large subject which can hardly be addressed here, but much of the impetus for its growth in recent decades has come from applications in optimization. An important reason is the fact that when a convex function is minimized over a convex set every locally optimal solution is global. Also, first-order necessary conditions for optimality turn out to be sufficient. A variety of other properties conducive to computation and interpretation of solutions ride on convexity as well. In fact the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity. Even for problems that aren't themselves of convex type, convexity may enter, for instance, in setting up subproblems as part of an iterative numerical scheme."

2.0 Notation

R: the set of all real numbers.

$\mathbf{R}_+$: the set of all nonnegative real numbers.

$\mathbf{R}_{++}$: the set of all positive real numbers.

$\mathbf{R}^p$: the set of p-dimensional real-valued (column) vectors.

$\mathbf{R}^{p\times q}$: the set of real-valued matrices of size p by q.

$\mathbf{S}^n$: the set of n by n symmetric matrices.

$\mathbf{S}^n_+$: the set of n by n symmetric positive semidefinite matrices.

$\mathbf{S}^n_{++}$: the set of n by n symmetric positive definite matrices.

dom f : domain where function $f(\boldsymbol{x})$ is defined.

2.1 Convex Sets and Convex Functions

Definition 2.1 A set $\mathcal{R}_c \subset E^n$ is said to be *convex* if for every pair of points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ in $\mathcal{R}_c$ and for every real number α in the range $0 < \alpha < 1$, the point $\boldsymbol{x} = \alpha\boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2$ is located in $\mathcal{R}_c$.

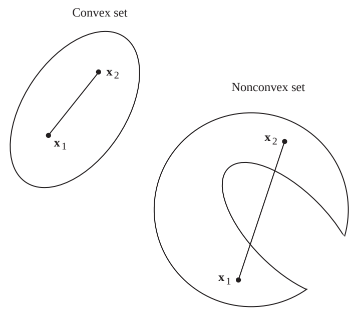


Figure 2.1 Convex and nonconvex sets.

Definition 2.2 A function $f(\boldsymbol{x})$ defined over a convex set $\mathcal{R}_c$ is said to be convex if for every pair of points in $\mathcal{R}_c$ and every real number α in the range 0 < α < 1, the following inequality holds

$$f[\alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2] \le \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2) \tag{2.1}$$

Geometrically, (2.1) says that $f(\boldsymbol{x})$ is convex if and only if the function's graph always lies underneath (or coincides with) the corresponding line segment, see Fig. 2.2.

- $f(\boldsymbol{x})$ is said to be *strictly* convex if (2.1) holds strictly, i.e.,

$$f[\alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2] < \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2)$$

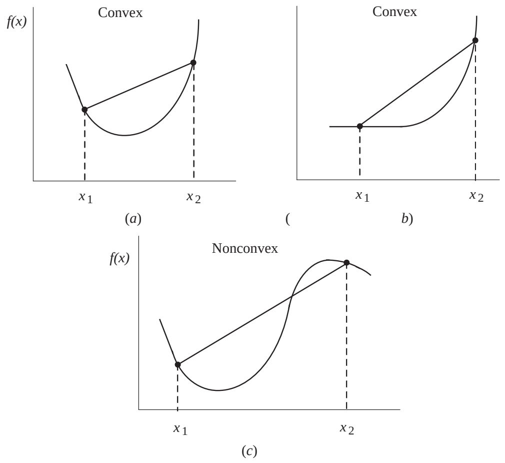


Figure 2.2 Convex and nonconvex functions

Definition 2.3 A function $f(\boldsymbol{x})$ is said to be concave if for every pair of points and every real number α in the range 0 < α < 1, the following inequality holds

$$f[\alpha \boldsymbol{x}_1 + (1-\alpha)\boldsymbol{x}_2] \ge \alpha f(\boldsymbol{x}_1) + (1-\alpha) f(\boldsymbol{x}_2) \tag{2.2}$$

- $f(\boldsymbol{x})$ is concave if and only if the function's graph always lies above (or coincides with) the corresponding line segment.
- $f(\boldsymbol{x})$ is said to be *strictly* concave if the inequality in (2.2) holds strictly.
- $f(\boldsymbol{x})$ is (strictly) convex if and only if $-f(\boldsymbol{x})$ is (strictly) concave, and vice versa.

Definition 2.4 The extended-value extension of a convex function $f(\boldsymbol{x})$ is defined as

$$\tilde{f}(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \boldsymbol{x} \in \operatorname{dom} f \\ +\infty & \boldsymbol{x} \notin \operatorname{dom} f \end{cases} \tag{2.3}$$

Examples of convex functions of one variable:

- affine function $f(x) = ax + b$
- exponential function $f(x) = e^{\alpha x}$ for any $\alpha \in \mathrm{R}$
- power function $f(x) = x^{\alpha}$ on R_{++} for $\alpha \ge 1$ or $\alpha \le 0$
- negative entropy $f(x) = x \log x$ on R_{++}

Examples of concave functions of one variable:

- affine function $f(x) = ax + b$
- logrithmic function $f(x) = \log x$ on R_{++}
- power function $f(x) = x^{\alpha}$ on R_{++} for $0 \le \alpha \le 1$

Examples of multivariable convex functions:

- affine function $f(\boldsymbol{x}) = \boldsymbol{a}^T \boldsymbol{x} + b$
- norm $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_p = \left(|x_1|^p + |x_2|^p + \cdots + |x_n|^p\right)^{1/p}$ for $p \ge 1$, and $\|\boldsymbol{x}\|_\infty = \max\limits_{1 \le i \le n}\{|x_i|\}$

2.2 Properties of Convex Functions

(1) ***Convexity of linear combination of convex functions*** If $f_1(\boldsymbol{x})$ and $f_2(\boldsymbol{x})$ are convex and a and b are nonnegative scalars, then $a f_1(\boldsymbol{x}) + b f_2(\boldsymbol{x})$ is convex.

(2) ***Relation between a convex function and convex sets*** If $f(\boldsymbol{x})$ is convex on a convex set $\mathcal{R}_c$, then the set $\mathcal{S}_K = \{\boldsymbol{x} : \boldsymbol{x} \in \mathcal{R}_c, f(\boldsymbol{x}) \le K\}$ is convex for every real number K. See Fig. 2.3 for an illustration of this property.

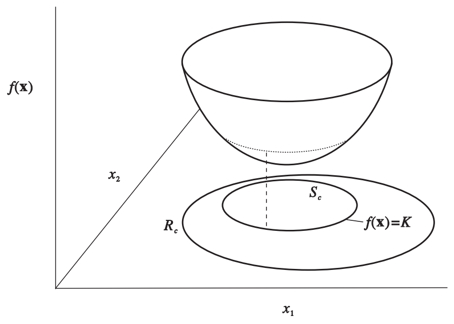


Figure 2.3

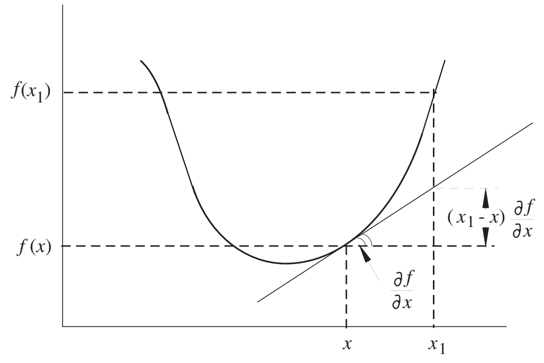
Convex functions can be characterized in different ways. The next definition of convexity turns out to be very useful.

(3) ***Property of convex function relating to gradient***

Suppose $f(\boldsymbol{x})$ is a smooth (C^1) function. Then $f(\boldsymbol{x})$ is convex over $\mathcal{R}_c$ if and only if

$$f(\boldsymbol{x}_1) \ge f(\boldsymbol{x}) + \nabla^T f(\boldsymbol{x})(\boldsymbol{x}_1 - \boldsymbol{x}) \tag{2.4}$$

for all $\boldsymbol{x}$ and $\boldsymbol{x}_1$ in $\mathcal{R}_c$. See Fig. 2.4 for an illustration of this property.



$f(x_1)$
$f(x)$
$(x_1 - x)\,\frac{\partial f}{\partial x}$
$\frac{\partial f}{\partial x}$
x
x_1

Fig. 2.4

(4) ***Property of convex functions relating to the Hessian*** A function $f(\boldsymbol{x}) \in C^2$ is convex over a convex $\mathcal{R}_c$ if and only if the Hessian $\nabla^2 f(\boldsymbol{x})$ is positive semidefinite over $\mathcal{R}_c$.

To prove property 4, we use Taylor expansion to write

$$f(\boldsymbol{x}_1) = f(\boldsymbol{x}) + \nabla^T f(\boldsymbol{x})(\boldsymbol{x}_1 - \boldsymbol{x}) + \tfrac{1}{2}(\boldsymbol{x}_1 - \boldsymbol{x})^T \nabla^2 f(\boldsymbol{x})(\boldsymbol{x}_1 - \boldsymbol{x}) + o(\|\boldsymbol{x}_1 - \boldsymbol{x}\|^2)$$

for all $\boldsymbol{x}$ and $\boldsymbol{x}_1$ in $\mathcal{R}_c$. Hence

$$f(\boldsymbol{x}_1) - f(\boldsymbol{x}) - \nabla^T f(\boldsymbol{x})(\boldsymbol{x}_1 - \boldsymbol{x}) = \tfrac{1}{2}(\boldsymbol{x}_1 - \boldsymbol{x})^T \nabla^2 f(\boldsymbol{x})(\boldsymbol{x}_1 - \boldsymbol{x}) + o(\|\boldsymbol{x}_1 - \boldsymbol{x}\|^2)$$

which implies that the left-hand side of the above equation is nonnegative if and only if $\nabla^2 f(\boldsymbol{x})$ is positive semidefinite. But from property 3 the left-hand side of the above equation is nonnegative if and only if $f(\boldsymbol{x})$ is convex. ■

(5) ***From an optimization perspective, what* (2.4) *tells us?*** (2.4) is a characterization of a function being convex, but it also provides a global linear lower bound of a convex that is tightest at a given point in the function's (convex) domain. The observation made below offers yet another way to appreciate the importance of convexity.

Let $\boldsymbol{x}$ be a point in the domain (let us assume it is the entire space R^n) of a smooth convex function $f(\boldsymbol{x})$ with gradient $\nabla f(\boldsymbol{x}) \ne \boldsymbol{0}$ (this is reasonable to assume. In fact, if $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$, then $\boldsymbol{x}$ would be a global minimizer of $f(\boldsymbol{x})$, hence not much left for us to do, see the next property). The hyperplane that contains point $\boldsymbol{x}$ and with gradient $\nabla f(\boldsymbol{x})$ as its normal can be described as a set of $\boldsymbol{x}_1$, each satisfies

$$\mathcal{P}_x: \quad \nabla^T f(\boldsymbol{x})(\boldsymbol{x}_1 - \boldsymbol{x}) = 0$$

and hyperplane $\mathcal{P}_x$ cuts (divides) the entire domain into two parts, see Fig. 2.5. Let us examine the part of the domain that contains gradient $\nabla f(\boldsymbol{x})$ (point $\boldsymbol{x} + \nabla f(\boldsymbol{x})$ to be precise). From the figure it is immediate that at each point $\boldsymbol{x}_1$ in this part (excluding hyperplane $\mathcal{P}_x$ itself) satisfies

$$\nabla^T f(\boldsymbol{x})(\boldsymbol{x}_1 - \boldsymbol{x}) > 0$$

which in conjunction with (2.4) gives

$$f(\boldsymbol{x}_1) \geq f(\boldsymbol{x}) + \nabla^T f(\boldsymbol{x})(\boldsymbol{x}_1 - \boldsymbol{x}) > f(\boldsymbol{x})$$

Therefore, this part of the domain contains *no* minimizers. In other words, the minimizers are all contained in the other part of the domain. We see that (2.4) is of help in reducing a region in searching a minimizer.

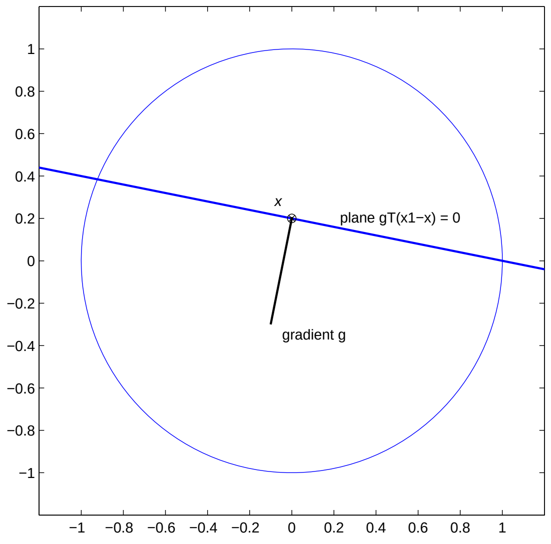


Figure 2.5

(6) ***Global solution of an unconstrained convex minimization problem***

Let $f(\boldsymbol{x})$ be a smooth convex function. If $\nabla f(\boldsymbol{x}^*) = \boldsymbol{0}$, then $\boldsymbol{x}^*$ is a global minimizer of $f(\boldsymbol{x})$. This property is an immediate consequence of (2.4):

$$f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) + \nabla^T f(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*) = f(\boldsymbol{x}^*)$$

(7) ***Globalness and convexity of minimizers in convex programming (CP) problems.*** Recall a constrained problem is a CP problem if it assumes the form

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to:} \ \boldsymbol{a}_i^T \boldsymbol{x} = b_i \quad \text{for } 1 \leq i \leq p \\ &\qquad\qquad\quad c_j(\boldsymbol{x}) \leq 0 \ \ \text{for } 1 \leq j \leq q \end{aligned}$$

where $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$ are convex. That is, a CP problem minimizes a *convex objective* *function* over *a convex feasible region*. We can state that

(a) $\boldsymbol{x}^*$ is a minimizer if and only if

$$\nabla f(\boldsymbol{x}^*)^T(\boldsymbol{x}-\boldsymbol{x}^*) \ge 0 \quad \text{for any feasible } \boldsymbol{x} \tag{2.5}$$

Proof The necessity of (2.5) immediately follows from Theorem 1.1, and the sufficiency of (2.5) follows from (2.4), namely,

$$f(\boldsymbol{x}) \ge f(\boldsymbol{x}^*) + \nabla^T f(\boldsymbol{x}^*)(\boldsymbol{x}-\boldsymbol{x}^*)$$

which in conjuction with (2.5) gives

$$f(\boldsymbol{x}) - f(\boldsymbol{x}^*) \ge \nabla^T f(\boldsymbol{x}^*)(\boldsymbol{x}-\boldsymbol{x}^*) \ge 0$$

hence $f(\boldsymbol{x}) \ge f(\boldsymbol{x}^*)$. ■

(b) If $\boldsymbol{x}^*$ is a local minimizer, then $\boldsymbol{x}^*$ is also a global minimizer.

(c) The set of minimizers of a CP problem is a convex set.

(d) If the objective function is strictly convex, then the minimizer is unique.

(8) ***Sufficiency of KKT Conditions for Convex Problems*** If $\boldsymbol{x}^*$ is a regular point of the constraints in (1.44) and satisfies the KKT conditions, then it is a global minimizer.

Proof For a feasible point $\bar{\boldsymbol{x}}$ with $\bar{\boldsymbol{x}} \ne \boldsymbol{x}^*$, we have $a_i(\bar{\boldsymbol{x}}) = 0$ for $1 \le i \le p$ and $c_j(\bar{\boldsymbol{x}}) \le 0$ for $1 \le j \le q$. Thus we can write

$$f(\bar{\boldsymbol{x}}) \ge f(\bar{\boldsymbol{x}}) + \sum_{j=1}^{q} \mu_j^* c_j(\bar{\boldsymbol{x}}).$$

Since $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$ are convex, we have

$$f(\bar{\boldsymbol{x}}) \ge f(\boldsymbol{x}^*) + \nabla^T f(\boldsymbol{x}^*)(\bar{\boldsymbol{x}}-\boldsymbol{x}^*) \quad \text{and} \quad c_j(\bar{\boldsymbol{x}}) \ge c_j(\boldsymbol{x}^*) + \nabla^T c_j(\boldsymbol{x}^*)(\bar{\boldsymbol{x}}-\boldsymbol{x}^*)$$

Hence

$$f(\bar{\boldsymbol{x}}) \ge f(\boldsymbol{x}^*) + \nabla^T f(\boldsymbol{x}^*)(\bar{\boldsymbol{x}}-\boldsymbol{x}^*) + \sum_{j=1}^{q} \mu_j^* \nabla^T c_j(\boldsymbol{x}^*)(\bar{\boldsymbol{x}}-\boldsymbol{x}^*) + \sum_{j=1}^{q} \mu_j^* c_j(\boldsymbol{x}^*)$$

In the light of the complementarity conditions, the last term in the above inequality vanishes, hence we have

$$f(\bar{\boldsymbol{x}}) \ge f(\boldsymbol{x}^*) + \left[\nabla f(\boldsymbol{x}^*) + \sum_{j=1}^{q} \mu_j^* \nabla c_j(\boldsymbol{x}^*)\right]^T (\bar{\boldsymbol{x}}-\boldsymbol{x}^*) \tag{2.6}$$

Since $a_i(\bar{\boldsymbol{x}}) = a_i(\boldsymbol{x}^*) = 0$, we can write

$$0 = a_i(\bar{\boldsymbol{x}}) - a_i(\boldsymbol{x}^*) = \boldsymbol{a}_i^T(\bar{\boldsymbol{x}}-\boldsymbol{x}^*) = \nabla^T a_i(\boldsymbol{x}^*)(\bar{\boldsymbol{x}}-\boldsymbol{x}^*)$$

Multiplying the above by λ_i^* and adding it to (2.6) for $1 \le i \le p$ gives

$$f(\bar{\boldsymbol{x}}) \ge f(\boldsymbol{x}^*) + \left[\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{p} \lambda_i^* \nabla a_i(\boldsymbol{x}^*) + \sum_{j=1}^{q} \mu_j^* \nabla c_j(\boldsymbol{x}^*) \right]^T (\bar{\boldsymbol{x}} - \boldsymbol{x}^*)$$

By the KKT conditions, the last term in the above inequality is zero, which leads to $f(\bar{\boldsymbol{x}}) \ge f(\boldsymbol{x}^*)$. This shows that $\boldsymbol{x}^*$ is a global minimizer. ■

2.3 How to Verify Convexity of a Function?

(1) By definitions. See Eqs. (2.1) and (2.4).

(2) Check if its Hessian is positive semidefinite, see Perperty (4) in Sec. 2.2.

(3) Check the function's convexity along lines: $f(\boldsymbol{x})$ is convex if and only if $g(t) = f(\boldsymbol{x} + t\boldsymbol{v})$ is convex in t in $\text{dom } g = \{t : \boldsymbol{x} + t\boldsymbol{v} \in \text{dom } f\}$ for any $\boldsymbol{x} \in \text{dom } f$.

- Example: $f(\boldsymbol{X}) = -\log\det(\boldsymbol{X})$ on $\text{dom} f = \text{S}_{++}^n$ is convex.

(4) Composition with affine function: if $f(\boldsymbol{x})$ is convex, then so is $f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$.

- Examples:

  (i) log barrier for linear inequalities, i.e.,

  $f(\boldsymbol{x}) = -\sum_{i=1}^{m} \log(b_i - \boldsymbol{a}_i^T \boldsymbol{x})$, $\text{dom} f = \{\boldsymbol{x} : \boldsymbol{a}_i^T \boldsymbol{x} < b_i, i = 1, 2, ..., m\}$ is convex;

  (ii) any norm of affine function $f(\boldsymbol{x}) = \| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} \|$ is convex.

(5) If all $f_i(\boldsymbol{x})$ are convex, then $f(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})\}$ is convex.

- Example: piecewise-linear function $f(\boldsymbol{x}) = \max\limits_{1 \le i \le m}\{\boldsymbol{a}_i^T \boldsymbol{x} + b_i\}$ is convex.

(6) If $g(\boldsymbol{x})$ is convex, $h(\boldsymbol{x})$ is convex and $\tilde{h}(\boldsymbol{x})$ is nondecreasing, then $f(\boldsymbol{x}) = h(g(\boldsymbol{x}))$ is convex. (here $\tilde{h}(\boldsymbol{x})$ denotes the extended-value extension of $h(\boldsymbol{x})$, see Definition 2.4)

- Example: if $f(\boldsymbol{x})$ is convex, then so is $e^{f(x)}$.

(7) If $g(\boldsymbol{x})$ is concave, $h(\boldsymbol{x})$ is convex and $\tilde{h}(\boldsymbol{x})$ is nonincreasing, then $f(\boldsymbol{x}) = h(g(\boldsymbol{x}))$ is convex.

- Examples:

  (i) if $f(\boldsymbol{x})$ is concave and positive, then $1/f(\boldsymbol{x})$ is convex.

(ii) if $f(\boldsymbol{x})$ is concave and positive, then $-\log f(\boldsymbol{x})$ is convex.

(8) A vector version of property (6): If all $g_i(\boldsymbol{x})$ are convex and $h(\boldsymbol{x}) = h(x_1, x_2, ..., x_m)$ is convex and $\tilde{h}(\boldsymbol{x})$ is nondecreasing in each variable, then $f(\boldsymbol{x}) = h(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), ..., g_m(\boldsymbol{x}))$ is convex.

- Example: if all $f_i(\boldsymbol{x})$ are convex, then $\log \sum_{i=1}^{m} e^{f_i(x)}$ is convex (Prob. 2.5).

(9) A vector version of property (7): If all $g_i(\boldsymbol{x})$ are concave and $h(\boldsymbol{x}) = h(x_1, x_2, ..., x_m)$ is convex and $\tilde{h}(\boldsymbol{x})$ is nonincreasing in each variable, then $f(\boldsymbol{x}) = h(g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), ..., g_m(\boldsymbol{x}))$ is convex.

- Example: if all $f_i(\boldsymbol{x})$ are concave and positive, then $-\sum_{i=1}^{m} \log f_i(\boldsymbol{x})$ is convex.

2.4 Duality

The concept of duality as applied to optimization is essentially a problem transformation that leads to an indirect but sometimes more efficient solution method. In a duality-based method, the original problem, which is referred to as the *primal* problem, is transformed into a problem in which the parameters are the Lagrange multipliers of the primal. The transformed problem is called the *dual* problem. In the case where the number of inequality constraints is much greater than the dimension of ***x***, solving the dual problem to find the Lagrange multipliers and then finding $\boldsymbol{x}^*$ for the primal problem becomes an attractive alternative.

2.4.1 The Lagrange Dual of a Convex Programming Problem

In this section we introduce another concept known as the Lagrange dual which turns out to be more useful in the study of convex programming. To this end we need a related concept called Lagrange dual function. Consider the general convex programming (CP) problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to:} \ \boldsymbol{a}_i^T \boldsymbol{x} = b_i \ \text{ for } 1 \le i \le p \\ &\qquad\qquad\ c_j(\boldsymbol{x}) \le 0 \ \text{ for } 1 \le j \le q \end{aligned} \tag{2.7}$$

and recall its Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i \left(\boldsymbol{a}_i^T \boldsymbol{x} - b_i \right) + \sum_{j=1}^{q} \mu_i \, c_j(\boldsymbol{x})$$

Definition 2.5 The *Lagrange dual function* of problem is defined as

$$q(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.8}$$

for $\boldsymbol{\lambda} \in R^p$ and $\boldsymbol{\mu} \in R^q$ with $\boldsymbol{\mu} \ge \boldsymbol{0}$. Note that the Lagrangian is *convex* with respect to ***x***. If $L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ is unbounded below for some $\{\boldsymbol{\lambda}, \ \boldsymbol{\mu}\}$, then the value of $q(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is assigned to $-\infty$.

Property 1 $q(\boldsymbol{\lambda},\boldsymbol{\mu})$ is a concave function with respect to $\{\boldsymbol{\lambda},\ \boldsymbol{\mu}\}$. The property follows from that fact that for $\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2 \in R^p$ and $\boldsymbol{\mu}_1,\boldsymbol{\mu}_2 \in R^q$ with $\boldsymbol{\mu}_1,\boldsymbol{\mu}_2 \ge \mathbf{0}$ and for $t \in (0,1)$, we have

$$q(t\boldsymbol{\lambda}_1+(1-t)\boldsymbol{\lambda}_2, t\boldsymbol{\mu}_1+(1-t)\boldsymbol{\mu}_2) = \inf_{\boldsymbol{x}} L(\boldsymbol{x}, t\boldsymbol{\lambda}_1+(1-t)\boldsymbol{\lambda}_1, t\boldsymbol{\mu}_1+(1-t)\boldsymbol{\mu}_2)$$

$$= \inf_{\boldsymbol{x}}\left[(t+1-t)f(\boldsymbol{x}) + \sum_{i=1}^{p}(t\lambda_{1,i}+(1-t)\lambda_{2,i})(a_i^T\boldsymbol{x}-b_i) + \sum_{j=1}^{q}(t\mu_{1,j}+(1-t)\mu_{2,j})c_j(\boldsymbol{x})\right]$$

$$\ge t\cdot\inf_{\boldsymbol{x}}\left[f(\boldsymbol{x})+\sum_{i=1}^{p}\lambda_{1,i}(a_i^T\boldsymbol{x}-b_i)+\sum_{j=1}^{q}\mu_{1,j}c_j(\boldsymbol{x})\right]+(1-t)\cdot\inf_{\boldsymbol{x}}\left[f(\boldsymbol{x})+\sum_{i=1}^{p}\lambda_{2,i}(a_i^T\boldsymbol{x}-b_i)+\sum_{j=1}^{q}\mu_{2,j}c_j(\boldsymbol{x})\right]$$

$$= t\cdot q(\boldsymbol{\lambda}_1,\boldsymbol{\mu}_1)+(1-t)\cdot q(\boldsymbol{\lambda}_2,\boldsymbol{\mu}_2)$$

Definition 2.6 The *Lagrange dual problem* with respect to problem (2.7) is defined as

$$\begin{aligned}&\underset{\lambda,\mu}{\text{maximize}} \quad q(\boldsymbol{\lambda},\boldsymbol{\mu})\\ &\text{subject to: } \boldsymbol{\mu}\ge\mathbf{0}\end{aligned} \tag{2.9}$$

Property 2 For any $\boldsymbol{x}$ feasible for problem (2.7) and $\{\boldsymbol{\lambda},\ \boldsymbol{\mu}\}$ feasible for problem (2.9), we have

$$f(\boldsymbol{x}) \ge q(\boldsymbol{\lambda},\boldsymbol{\mu}) \tag{2.10}$$

This is because

$$L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}) = f(\boldsymbol{x})+\sum_{i=1}^{p}\lambda_i\left(\boldsymbol{a}_i^T\boldsymbol{x}-b_i\right)+\sum_{j=1}^{q}\mu_i\,c_j(\boldsymbol{x}) = f(\boldsymbol{x})+\sum_{j=1}^{q}\mu_i\,c_j(\boldsymbol{x}) \le f(\boldsymbol{x})$$

thus

$$q(\boldsymbol{\lambda},\boldsymbol{\mu}) = \inf_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}) \le L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}) \le f(\boldsymbol{x})$$

We call the convex minimization problem in (2.7) the *primal problem* and the concave maximization problem in (2.9) the *dual problem*. From (2.10), it is natural to introduce a *duality gap* between the primal and dual objectives as

$$\delta(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}) = f(\boldsymbol{x}) - q(\boldsymbol{\lambda},\boldsymbol{\mu}) \tag{2.11}$$

It follows that for feasible $\{\boldsymbol{x},\boldsymbol{\lambda},\ \boldsymbol{\mu}\}$ the duality gap is always nonnegative.

Property 3 Let $\boldsymbol{x}^*$ be a solution of the primal problem in (2.7). Then the dual function at any feasible $\{\boldsymbol{\lambda},\ \boldsymbol{\mu}\}$ serves as a lower bound of the optimal value of the primal objective, $f(\boldsymbol{x}^*)$, namely,

$$f(\boldsymbol{x}^*) \ge q(\boldsymbol{\lambda},\boldsymbol{\mu}) \tag{2.12}$$

This property follows immediately from (2.10) by taking the minimum of $f(\boldsymbol{x})$ on its left-hand side.

- Now a question naturally arises: what will be the tightest lower bound the dual function $q(\boldsymbol{\lambda},\boldsymbol{\mu})$ can offer? From (2.12), obviously the answer is found by maximizing the dual function $q(\boldsymbol{\lambda},\boldsymbol{\mu})$ on the right-hand side of (2.12) subject to $\boldsymbol{\mu} \ge \mathbf{0}$. This is exactly the Lagrange dual problem as formulated in (2.9). Therefore, if we denote the solution of (2.9) by $(\boldsymbol{\lambda}^*,\ \boldsymbol{\mu}^*)$, then we have

$$f(\boldsymbol{x}^*) \ge q(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*) \tag{2.13}$$

Based on (2.13), we now introduce the concept of *strong* and *weak* duality as follows.

Definition 2.7 Let $\boldsymbol{x}^*$ and $(\boldsymbol{\lambda}^*,\ \boldsymbol{\mu}^*)$ be solutions of primal problem (2.7) and dual problem (2.9), respectively. We say strong duality holds if $f(\boldsymbol{x}^*) = q(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*)$, i.e., the optimal duality gap is zero; and a weak duality holds if $f(\boldsymbol{x}^*) > q(\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*)$.

- It can be shown that if the primal problem is strictly feasible, i.e., there exists $\boldsymbol{x}$ satisfying

$$\begin{aligned} \boldsymbol{a}_i^T\boldsymbol{x} &= b_i \quad \text{for } 1 \le i \le p \\ c_j(\boldsymbol{x}) &< 0 \quad \text{for } 1 \le j \le q \end{aligned}$$

(this is to say that the interior of the feasible region of problem (2.7) is nonempty), then strong duality holds, i.e., the optimal duality gap is zero.

Example 2.1 Find the Lagrange dual of the LP problem

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{c}^T\boldsymbol{x} \\ &\text{subject to: } \boldsymbol{Ax} = \boldsymbol{b},\ \boldsymbol{x} \ge \mathbf{0} \end{aligned} \tag{1.3}$$

Solution We write $\boldsymbol{x} \ge \mathbf{0}$ as $-\boldsymbol{x} \le \mathbf{0}$ hence the Lagrangian of the LP problem is given by

$$L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}) = \boldsymbol{c}^T\boldsymbol{x} + (\boldsymbol{Ax}-\boldsymbol{b})^T\boldsymbol{\lambda} - \boldsymbol{x}^T\boldsymbol{\mu}$$

thus

$$q(\boldsymbol{\lambda},\boldsymbol{\mu}) = \inf_{\boldsymbol{x}}\left\{\boldsymbol{c}^T\boldsymbol{x} + (\boldsymbol{Ax}-\boldsymbol{b})^T\boldsymbol{\lambda} - \boldsymbol{x}^T\boldsymbol{\mu}\right\} = \inf_{\boldsymbol{x}}\left\{(\boldsymbol{c}+\boldsymbol{A}^T\boldsymbol{\lambda}-\boldsymbol{\mu})^T\boldsymbol{x} - \boldsymbol{b}^T\boldsymbol{\lambda}\right\} \tag{2.14}$$

For given $\{\boldsymbol{\lambda},\boldsymbol{\mu}\}$ such that $\boldsymbol{c}+\boldsymbol{A}^T\boldsymbol{\lambda}-\boldsymbol{\mu} \ne \mathbf{0}$, from (2.14) we have $q(\boldsymbol{\lambda},\boldsymbol{\mu}) = -\infty$. Therefore, to deal with a well-defined dual function $q(\boldsymbol{\lambda},\boldsymbol{\mu})$ we assume $\boldsymbol{c}+\boldsymbol{A}^T\boldsymbol{\lambda}-\boldsymbol{\mu} = \mathbf{0}$ which leads to

$$q(\boldsymbol{\lambda},\boldsymbol{\mu}) = \inf_{\boldsymbol{x}}(-\boldsymbol{b}^T\boldsymbol{\lambda}) = -\boldsymbol{b}^T\boldsymbol{\lambda}$$

and the Lagrange dual of (1.3) is given by

$$\begin{array}{ll} \underset{\boldsymbol{\lambda},\boldsymbol{\mu}}{\text{maximize}} & -\boldsymbol{b}^T\boldsymbol{\lambda} \\ \text{subject to:} & \boldsymbol{\mu} \ge \boldsymbol{0} \end{array}$$

Since $\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda} - \boldsymbol{\mu} = \boldsymbol{0}$, $\boldsymbol{\mu} = \boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}$ so the above problem becomes

$$\begin{array}{ll} \underset{\boldsymbol{\lambda}}{\text{maximize}} & -\boldsymbol{b}^T\boldsymbol{\lambda} \\ \text{subject to:} & -\boldsymbol{c} - \boldsymbol{A}^T\boldsymbol{\lambda} \le \boldsymbol{0} \end{array}$$

which is equivalent to

$$\begin{array}{ll} \underset{\boldsymbol{\lambda}}{\text{minimize}} & \boldsymbol{b}^T\boldsymbol{\lambda} \\ \text{subject to:} & (-\boldsymbol{A}^T)\boldsymbol{\lambda} \le \boldsymbol{c} \end{array}$$ ■

Example 2.2 Find the Lagrange dual of the QP problem

$$\begin{array}{ll} \text{minimize} & \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{p}^T\boldsymbol{x} \\ \text{subject to:} & \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \end{array} \tag{1.5}$$

where $\boldsymbol{H}$ is positive definite.

Solution The Lagrangian of the QP problem is given by

$$L(\boldsymbol{x},\boldsymbol{\mu}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{p}^T\boldsymbol{x} + \boldsymbol{\mu}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$$

Hence
$$q(\boldsymbol{\mu}) = \inf_{\boldsymbol{x}}\left\{\tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{p}^T\boldsymbol{x} + \boldsymbol{\mu}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})\right\} \tag{2.15}$$

where the infimum is attained at $\boldsymbol{x} = -\boldsymbol{H}^{-1}(\boldsymbol{p} + \boldsymbol{A}^T\boldsymbol{\mu})$. By substituting this solution into (2.15), we obtain

$$q(\boldsymbol{\mu}) = -\tfrac{1}{2}\boldsymbol{\mu}^T\boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{A}^T\boldsymbol{\mu} - \boldsymbol{\mu}^T(\boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{p} + \boldsymbol{b}) - \tfrac{1}{2}\boldsymbol{p}^T\boldsymbol{H}^{-1}\boldsymbol{p} \tag{2.16}$$

If we let $\boldsymbol{P} = \boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{A}^T$, $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{p} + \boldsymbol{b}$ and neglect the constant term in (2.16), the dual function becomes $q(\boldsymbol{\mu}) = -\tfrac{1}{2}\boldsymbol{\mu}^T\boldsymbol{P}\boldsymbol{\mu} - \boldsymbol{\mu}^T\boldsymbol{t}$, hence the Lagrange dual of (1.5) is given by

$$\begin{array}{ll} \text{minimize} & \tfrac{1}{2}\boldsymbol{\mu}^T\boldsymbol{P}\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{t} \\ \text{subject to:} & \boldsymbol{\mu} \ge \boldsymbol{0} \end{array} \tag{2.17}$$

Note that by definition matrix $\boldsymbol{P}$ in (2.17) is positive definite, therefore the dual problem is also a convex QP problem, but with simpler constraints in comparison with the primal problem in (1.5). In addition, if the number of constraints involved in the primal problem is smaller than *n*, so is the size of the dual problem. ■

Problems

2.1 Verify the convexity of the following one-variable functions:

(a) $f(x) = e^x$

(b) $f(x) = |x|^p$ with $p > 1$

(c) $f(x) = \dfrac{x^2}{1-|x|}$

(d) $f(x) = |x| - \log(1+|x|)$

2.2 The l_p norm (with $p > 1$) of a vector $\boldsymbol{x}$ is defined by $\|\boldsymbol{x}\|_p = \left[\sum_{i=1}^n |x_i|^p\right]^{1/p}$. Show that $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_p$ with $p > 1$ is a convex function.

2.3 Let Q be an arbitrary set. Show that function $\psi_Q(\boldsymbol{x}) = \sup\limits_{\boldsymbol{g} \in Q} \boldsymbol{g}^T \boldsymbol{x}$ is convex. (In the literature when Q is a convex set function $\psi_Q(\boldsymbol{x})$ is called *support* function of set Q)

2.4 Verify the convexity of the following functions.

(a) $f(\boldsymbol{x}) = \sum_{i=1}^n x_i \log x_i$ over domain $\text{dom}(f) = \{\boldsymbol{x} : x_i > 0 \text{ for } 1 \le i \le n\}$.

(b) $f(\boldsymbol{x}) = -\sum_{i=1}^m \log(b_i - \boldsymbol{a}_i^T \boldsymbol{x})$ over domain $\text{dom}(f) = \{\boldsymbol{x} : \boldsymbol{a}_i^T \boldsymbol{x} < b_i \text{ for } 1 \le i \le m\}$.

(c) *Max* function defined by $f(\boldsymbol{x}) = \max\{x_1, x_2, ..., x_n\}$ over domain R^n.

(d) *Log-sum-exp* function defined by $f(\boldsymbol{x}) = \log\left(\sum_{i=1}^n e^{x_i}\right)$.

(e) *Quadratic-over-linear* function defined by $f(x,y) = \dfrac{x^2}{y}$ on $\text{dom}(f) = \{(x,y) \in R^2, y > 0\}$

(f) *Perspective* of f defined by $h(\boldsymbol{x},t) = t \cdot f(\boldsymbol{x}/t)$ as a function of ($\boldsymbol{x}$, t) over domain $\text{dom}(f) = \{(\boldsymbol{x},t) : \boldsymbol{x} \in R^n, t > 0\}$, where $f(\boldsymbol{x})$ is convex.

2.5 Use the result from part (d) of Prob. 2.4 and property (8) in Sec. 2.3 to show that if all $f_i(\boldsymbol{x})$ are convex, then $f(\boldsymbol{x}) = \log \sum_{i=1}^m e^{f_i(\boldsymbol{x})}$ is convex.

2.6 Using the result from part (f) of Prob.. 2.4 to show that $h(x,t) = -t\log\left(1+\gamma\frac{x}{t}\right)$, where $t > 0$ and γ is a positive constant, as a function of (x, t) is convex.

2.7 Consider the McCormick function which is defined by

$$f(\boldsymbol{x}) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$$

over the region $-3 \le x_1, x_2 \le 4$. To investigate the convexity of the function over the region, place a uniform and dense grid, say 120 by 120, over the region and check whether or not the function is convex at each of these grid points.

(a) You can do so by creating a function which assumes value 1 if $f(\boldsymbol{x})$ is convex and 0 if $f(\boldsymbol{x})$ is not convex, and display the function so created using **`mesh`** with an appropriate view position.

(b) Visually verify the conclusion made in part (a) by plotting the McCormick function over the above region using **`mesh`**.

2.8 Suppose functions $f_i(\boldsymbol{x})$ for $i = 1, 2, \ldots, m$ are convex.

(a) Show that

$$\phi(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})\} \tag{P2.1}$$

is a convex function.

(b) Does the claim hold if m becomes $+\infty$? Prove or disprove it.

2.9 An *affine-max* function is defined by

$$f(\boldsymbol{x}) = \max\{\boldsymbol{a}_1^T\boldsymbol{x} + b_1,\ \boldsymbol{a}_2^T\boldsymbol{x} + b_2, \ldots, \boldsymbol{a}_m^T\boldsymbol{x} + b_m\} \tag{P2.2}$$

(a) Use a simple example with $m = 3$ to demonstrate $f(\boldsymbol{x})$ in (P2.2) is piecewise linear.

(b) Show that $f(\boldsymbol{x})$ is convex.

(c) Given data set $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_K, y_K)$, where $\boldsymbol{x}_i$ is an input to a system that produces output y_i. Suppose one selects a "model", which is a function, say $f(\boldsymbol{x}, \boldsymbol{p})$, of $\boldsymbol{x}$ and a parameter vector $\boldsymbol{p}$, to represent the system. A least-square (LS) data fitting problem is find an optimal parameter $\boldsymbol{p}^*$ that minimizes the error

$$J(\boldsymbol{p}) = \sum_{i=1}^{K}\left(f(\boldsymbol{x}_i, \boldsymbol{p}) - y_i\right)^2$$

Typically the size of data set, K, is large, and the idea of data fitting is to use small number of parameters (in the case of using $f(\boldsymbol{x}, \boldsymbol{p})$ in (P2.2) as the model function, this means that $m << K$)

to model a large data set as accurately as possible.
Follow the above to formulate a piecewise LS data fitting problem where function *f* is given by (P2.2). In the problem formulation, an expression of parameter vector $\boldsymbol{p}$ should be given explicitly.

2.10

(a) Show that convexity is an invariant property under affine transformation. This is to say that if $f(\boldsymbol{y})$ is convex function of $\boldsymbol{y} \in R^{m\times 1}$ and $\boldsymbol{A} \in R^{m\times n}$ and $\boldsymbol{b} \in R^{m\times 1}$, then $\phi(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b})$ is a convex function of $\boldsymbol{x} \in R^{n\times 1}$.

(b) Show that function

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} e^{a_i^T \boldsymbol{x} + \tau_i}$$

is convex. Hint: Use the results from Probs. 2.1(a) and 2.8(a).

2.11 Show that a continuously differentiable function $f(\boldsymbol{x})$ is convex if and only if for any $\boldsymbol{x}$ and $\boldsymbol{y}$ we have

$$(\boldsymbol{x}-\boldsymbol{y})^T(\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})) \ge 0$$

2.12 Show that a twice continuously differentiable function $f(\boldsymbol{x})$ is convex if and only if for any $\boldsymbol{x}$ the Hessian of $f(\boldsymbol{x})$ is positive semidefinite.

2.13 (*Jensen's inequality*) Show that if $f(\boldsymbol{x})$ is a convex function, then for $\sum_{i=1}^{m} \alpha_i = 1$ with all $\alpha_i \ge 0$, we have

$$f\left(\sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i\right) \le \sum_{i=1}^{m} \alpha_i f(\boldsymbol{x}_i) \tag{P2.3}$$

2.14 A point $\boldsymbol{x}$ is said to be a convex combination of points $\{\boldsymbol{x}_i, i = 1, 2, \ldots, m\}$ if $\boldsymbol{x}$ can be expressed as $\boldsymbol{x} = \sum_{i=1}^{m} \alpha_i \boldsymbol{x}_i$ where $\sum_{i=1}^{m} \alpha_i = 1$ with all $\alpha_i \ge 0$. Show that if $f(\boldsymbol{x})$ is convex, then

$$f(\boldsymbol{x}) \le \max_{1\le i\le m} f(\boldsymbol{x}_i)$$

2.15 Let dom(*f*) denote the *domain* of function $f(\boldsymbol{x})$ where $f(\boldsymbol{x})$ is defined. Show that $f(\boldsymbol{x})$ is convex if and only if its *epigraph* defined by

$$\text{epi}(f) = \{(\boldsymbol{x}, t) : \boldsymbol{x} \in \text{dom}(f) \text{ and } t \ge f(\boldsymbol{x})\} \tag{P2.4}$$

is a convex set. This is an important and useful result as it characterizes a convex function by a convex set.

2.16 Show that for a convex function $f(\boldsymbol{x})$, $\nabla f(\boldsymbol{x})$ defines a supporting hyperplane to epi(f) at $(\boldsymbol{x}, f(\boldsymbol{x}))$ by proving

$$\begin{bmatrix} \nabla f(\boldsymbol{x}) \\ -1 \end{bmatrix}^T \left(\begin{bmatrix} \boldsymbol{y} \\ t \end{bmatrix} - \begin{bmatrix} \boldsymbol{x} \\ f(\boldsymbol{x}) \end{bmatrix} \right) \le 0 \quad \forall\, (\boldsymbol{y}, t) \in \text{epi}(f) \qquad \text{(P2.5)}$$

and then illustrating the inequality in (P2.5) using a graph.

2.17 Study the convexity of the function given below

$f(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \ln\left(1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}_i}\right)$ where $\ln(\cdot)$ denotes natural logarithm, and {$\boldsymbol{x}_i$ for i = 1, 2, …, N} is a given data set.

2.18 Given the QP problem

$$\begin{aligned} &\text{minimize} \quad \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{x} \\ &\text{subject to:} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned}$$

where $\boldsymbol{A} \in R^{m\times n}$ has full (row) rank. Find the largest lower bound of the above objective function via its Lagrange dual.

2.19 Consider the *two-way partitioning* problem

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{x}^T\boldsymbol{W}\boldsymbol{x} \\ &\text{subject to:} \quad x_i \in \{1, -1\} \text{ for } i = 1, 2, ..., n \end{aligned}$$

where the (i, j)th component of $\boldsymbol{W}$, $w_{i,j}$, represents the cost of assigning i and j to the same set. Derive a reasonable lower bound of the above objective function via its Lagrange dual.

Chapter 3 LP, QP, SDP, SOCP Problems and Software

In this chapter we examine several specific classes of convex constrained problems. For each class, examples are provided to illustrate its usefulness by explaining how practical problems can be formulated as that type of optimization. A popular CP solver known as **CVX** is introduced to facilitate computing numerical solution of CP problems.

3.1 Linear Programming

3.1.1 Examples

Example 3.1 *Transportation problem*

Quantities $q_1, q_2, \ldots, q_m$ of a certain product are produced by m manufacturing divisions of a company, which are at distinct locations. The product is to be shipped to n destinations that require quantities $b_1, b_2, \ldots, b_n$. Assume that the cost of shipping a unit from manufacturing division i to destination j is $c_{i,j}$ with $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

Let $x_{i,j}$ to be the quantity shipped from division i to destination j so as to minimize the total cost of transportation. There are several constraints on variables $x_{i,j}$. First, each division can provide only a fixed quantity of the product, hence

$$\sum_{j=1}^{n} x_{i,j} = q_i \quad \text{for } i = 1, 2, \ldots, m$$

Second, the quantity to be shipped to a specific destination has to meet the need of that destination and so

$$\sum_{i=1}^{m} x_{i,j} = b_j \quad \text{for } j = 1, 2, \ldots, n$$

In addition, the variables $x_{i,j}$ are nonnegative and thus, we have

$$x_{i,j} \ge 0 \quad \text{for } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, n$$

The transportation problem is to find quantities $\{x_{i,j}\}$ that minimizes $C = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j}$ subject to the above constraints. If we let

$$\boldsymbol{c} = \begin{bmatrix} c_{11} & \cdots & c_{1n} & c_{21} & \cdots & c_{2n} & \cdots & c_{m1} & \cdots & c_{mn} \end{bmatrix}^T$$

$$\boldsymbol{x} = \begin{bmatrix} x_{11} & \cdots & x_{1n} & x_{21} & \cdots & x_{2n} & \cdots & x_{m1} & \cdots & x_{mn} \end{bmatrix}^T$$

$$\boldsymbol{A} = \begin{bmatrix} 1 & 1 & \cdots & 1 & 0 & 0 & \cdots & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 1 & \cdots & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & \cdots & 1 & 1 & \cdots & 1 \\ 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix}$$

$$\boldsymbol{b} = \begin{bmatrix} q_1 & \cdots & q_m & b_1 & \cdots & b_n \end{bmatrix}^T$$

then the transportation problem can be stated as

$$\begin{aligned} &\text{minimize} \quad C = \boldsymbol{c}^T \boldsymbol{x} \\ &\text{subject to:} \;\; \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\qquad\qquad\quad \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

Since both the objective function and constraints are linear, the problem is known as a *linear programming (LP) problem*. ■

Example 3.2 *Chebyshev centre of a polyhedron*

A polyhedron is a set defined by

$$\mathcal{P} = \{\boldsymbol{x} : \boldsymbol{a}_i^T \boldsymbol{x} \leq b_i, i = 1, 2, ..., q\}$$

The Chebeshev centre of $\mathcal{P}$ is the centre $\boldsymbol{x}_c$ of the largest ball $\mathcal{B} = \{\boldsymbol{x}_c + \boldsymbol{u} : \| \boldsymbol{u} \|_2 \leq r\}$ that is inscribed in $\mathcal{P}$.

To find the largest inscribed ball, the centre $\boldsymbol{x}_c$ and radius r of ball $\mathcal{B}$ are treated as *variables* such that r is to be maximized among the balls inscribed in $\mathcal{P}$. Note that

$$\boldsymbol{a}_i^T \boldsymbol{x} \leq b_i \text{ for } \boldsymbol{x} \in \mathcal{B} \text{ for } i = 1, 2, ..., q$$

$$\Updownarrow$$

$$\sup_{\|\boldsymbol{u}\|_2 \leq r} \{\boldsymbol{a}_i^T (\boldsymbol{x}_c + \boldsymbol{u}) \leq b_i\} \text{ for } i = 1, 2, ..., q$$

$$\Updownarrow$$

$$\boldsymbol{a}_i^T \boldsymbol{x}_c + \sup_{\|\boldsymbol{u}\|_2 \leq r} \{\boldsymbol{a}_i^T \boldsymbol{u}\} \leq b_i \text{ for } i = 1, 2, ..., q$$

$$\Updownarrow$$

$$\boldsymbol{a}_i^T \boldsymbol{x}_c + r \| \boldsymbol{a}_i \|_2 \leq b_i \text{ for } i = 1, 2, ..., q$$

Therefore, the problem at hand can be formulated as

$$\begin{aligned} &\text{minimize} \quad -r \\ &\text{subject to:} \;\; \boldsymbol{a}_i^T \boldsymbol{x}_c + r \| \boldsymbol{a}_i \|_2 \leq b_i \text{ for } i = 1, 2, ..., q \end{aligned}$$

which is obviously an LP problem. ■

Example 3.3 *Piecewise-linear minimization*

It is well known that the piecewise linear function $f(\boldsymbol{x}) = \max_{1 \leq i \leq q} \{\boldsymbol{a}_i^T \boldsymbol{x} + b_i\}$ itself is convex, and is a simple model to approximate sophisticated convex functions. Here we consider minimizing such a function, i.e.,

$$\text{minimize} \;\; \max_{1 \leq i \leq q} \{\boldsymbol{a}_i^T \boldsymbol{x} + b_i\}$$

By introducing an auxiliary scalar variable t, the above problem can be cast as an LP problem:

$$\begin{aligned} &\text{minimize} \quad t \\ &\text{subject to:} \;\; \boldsymbol{a}_i^T \boldsymbol{x} + b_i \leq t \;\text{ for }\; i = 1, 2, ..., q \end{aligned}$$

■

3.1.2 Primal and Dual of LP Problems

Consider the standard LP problem

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{c}^T \boldsymbol{x} \\ &\text{subject to:} \;\; \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\qquad\qquad\quad\;\; \boldsymbol{x} \ge \boldsymbol{0} \end{aligned} \tag{3.1}$$

where matrix $\boldsymbol{A} \in R^{p \times n}$ is of full row rank as the *primal* problem. From Chapter 2, the *dual* problem to (3.1) is given by

$$\begin{aligned} &\underset{\boldsymbol{\lambda},\,\boldsymbol{\mu}}{\text{maximize}} \quad -\boldsymbol{b}^T \boldsymbol{\lambda} \\ &\text{subject to:} \;\; -\boldsymbol{A}^T\boldsymbol{\lambda} + \boldsymbol{\mu} = \boldsymbol{c} \\ &\qquad\qquad\quad\;\; \boldsymbol{\mu} \ge \boldsymbol{0} \end{aligned} \tag{3.2}$$

It is known from the KKT conditions that $\boldsymbol{x}^*$ is a minimizer of the problem in (3.1) if and only if there exist $\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^*$ with $\boldsymbol{\mu}^* \ge \boldsymbol{0}$ such that

$$-\boldsymbol{A}^T\boldsymbol{\lambda}^* + \boldsymbol{\mu}^* = \boldsymbol{c} \tag{3.3a}$$

$$\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b} \tag{3.3b}$$

$$x_i^* \cdot \mu_i^* = 0 \quad \text{for} \;\; 1 \le i \le n \tag{3.3c}$$

$$\boldsymbol{x}^* \ge \boldsymbol{0}, \quad \boldsymbol{\mu}^* \ge \boldsymbol{0} \tag{3.3d}$$

It follows that $\boldsymbol{x}^*$ is a feasible point of (3.1) and $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is a feasible pair of (3.2). Actually, $\boldsymbol{x}^*$ solves problem (3.1) and $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ solves the dual problem (3.2). Therefore we shall call $\{\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*\}$ satisfying (3.3) a primal-dual solution.

Let $\{\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*\}$ satisfies (3.3), then $\boldsymbol{x*}$ is a solution of (3.1) and $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ are the Lagrange multipliers of (3.1). Furthermore, $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ is a solution of (3.2) and $\boldsymbol{x}^*$ may be interpreted as the Lagrange multiplier of the dual problem (3.2).

The *duality gap*, defined as the difference between the cost of the primal and the cost of the dual, is given by

$$\delta(\boldsymbol{x}, \boldsymbol{\lambda}) = \boldsymbol{c}^T\boldsymbol{x} - \left(-\boldsymbol{b}^T\boldsymbol{\lambda}\right) \tag{3.4}$$

which is always nonnegative, namely,

$$\delta(\boldsymbol{x}, \boldsymbol{\lambda}) \ge 0 \tag{3.5}$$

This is because from (3.1) and (3.2) we have

$$\boldsymbol{c}^T \boldsymbol{x} = \left(-\boldsymbol{A}^T \boldsymbol{\lambda} + \boldsymbol{\mu}\right)^T \boldsymbol{x} = -\boldsymbol{\lambda}^T \boldsymbol{A}\boldsymbol{x} + \boldsymbol{\mu}^T \boldsymbol{x} \ge -\boldsymbol{b}^T \boldsymbol{\lambda} \tag{3.6}$$

Furthermore, at the primal-dual solution $\{\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*\}$, the duality gap is reduced to its minimum – zero, i.e.,

$$\delta(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = 0 \tag{3.7}$$

This follows from a line of argument similar to (3.6) in conjunction with the complementarity condition (3.3c):

$$\boldsymbol{c}^T \boldsymbol{x}^* = \left(-\boldsymbol{A}^T \boldsymbol{\lambda}^* + \boldsymbol{\mu}^*\right)^T \boldsymbol{x}^* = -\boldsymbol{\lambda}^{*T} \boldsymbol{A}\boldsymbol{x}^* + \boldsymbol{\mu}^{*T} \boldsymbol{x}^* = -\boldsymbol{b}^T \boldsymbol{\lambda}^*$$

· Another important concept related to primal-dual solutions is *central path*. By (3.3), set $\{\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}\}$ is a primal-dual solution if it satisfies the conditions

$$\begin{aligned} &\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} && \text{with } \boldsymbol{x} \ge \boldsymbol{0} \\ &-\boldsymbol{A}^T \boldsymbol{\lambda} + \boldsymbol{\mu} = \boldsymbol{c} && \text{with } \boldsymbol{\mu} \ge \boldsymbol{0} \\ &\boldsymbol{X}\boldsymbol{\mu} = \boldsymbol{0} \end{aligned} \tag{3.8}$$

where $\boldsymbol{X} = \text{diag}\{x_1, x_2, \ldots, x_n\}$.

3.2 Quadratic Programming

General quadratic programming (QP) problems assume the form

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\boldsymbol{x}^T \boldsymbol{H}\boldsymbol{x} + \boldsymbol{p}^T \boldsymbol{x} + \kappa \\ &\text{subject to:} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & && \boldsymbol{C}\boldsymbol{x} \le \boldsymbol{d} \end{aligned} \tag{3.9a-c}$$

The QP problem is convex if matrix $\boldsymbol{H}$ is positive semidefinite.

Example 3.4 *Unconstrained least-squares (LS) problem*

$$\text{minimize } \tfrac{1}{2} \| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \|_2^2$$

where $\boldsymbol{A} \in R^{p \times n}$ with $p > n$. The objective function is quadratic and convex because

$$\tfrac{1}{2} \| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \|_2^2 = \tfrac{1}{2} \boldsymbol{x}^T \boldsymbol{A}^T \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{A}\boldsymbol{x} + \tfrac{1}{2} \| \boldsymbol{b} \|_2^2$$

If $\boldsymbol{A}$ has full (column) rank, the LS problem has unique and global solution $\boldsymbol{x}^* = \left(\boldsymbol{A}^T \boldsymbol{A}\right)^{-1} \boldsymbol{A}^T \boldsymbol{b}$.

■

Example 3.5 *QP with equality constraints*

Consider convex QP problem

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\boldsymbol{x}^T \boldsymbol{H}\boldsymbol{x} + \boldsymbol{p}^T \boldsymbol{x} + \kappa \\ &\text{subject to:} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{3.10a-b}$$

where $\boldsymbol{A} \in R^{p\times n}$. An effective approach to solve (3.10) is to use the QR decomposition of $\boldsymbol{A}^T$, i.e.,

$$\boldsymbol{A}^T = \boldsymbol{Q}\begin{bmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{bmatrix} \tag{3.11}$$

where $\boldsymbol{Q}$ is an $n\times n$ orthogonal and $\boldsymbol{R}$ is a $p\times p$ upper triangular matrix (see Appendix). Using (3.11), the constraints in (3.10b) can be expressed as

$$\boldsymbol{R}^T \hat{\boldsymbol{x}}_1 = \boldsymbol{b}$$

where $\hat{\boldsymbol{x}}_1$ is the vector composed of the first p elements of $\hat{\boldsymbol{x}} = \boldsymbol{Q}^T\boldsymbol{x}$. If we denote

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{\boldsymbol{x}}_1 \\ \boldsymbol{\phi} \end{bmatrix} \text{ and } \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix}$$

with $\boldsymbol{\phi} \in R^{(n-p)\times 1}$, $\boldsymbol{Q}_1 \in R^{n\times p}$, and $\boldsymbol{Q}_2 \in R^{n\times(n-p)}$, then we obtain

$$\boldsymbol{x} = \boldsymbol{Q}_2\boldsymbol{\phi} + \boldsymbol{Q}_1\boldsymbol{R}^{-T}\boldsymbol{b} \tag{3.12}$$

The parameterized solutions in (3.12) can be used to convert the problem in (3.10) into a unconstrained problem which leads to the unique global minimizer of the problem in (3.10) as

$$\boldsymbol{x}^* = \boldsymbol{Q}_2\boldsymbol{\phi}^* + \boldsymbol{Q}_1\boldsymbol{R}^{-T}\boldsymbol{b} \tag{3.13a}$$

where $\boldsymbol{\phi}^*$ is the solution of the linear system

$$\left(\boldsymbol{Q}_2^T\boldsymbol{H}\boldsymbol{Q}_2\right)\boldsymbol{\phi} = -\boldsymbol{Q}_2^T\left(\boldsymbol{H}\boldsymbol{Q}_1\boldsymbol{R}^{-T}\boldsymbol{b} + \boldsymbol{p}\right) \tag{3.13b}$$

Alternatively, problem (3.10) can also be solved by using the first-order necessary conditions described in Theorem 1.1, which are given by

$$\begin{aligned} &\boldsymbol{H}\boldsymbol{x}^* + \boldsymbol{p} + \boldsymbol{A}^T\boldsymbol{\lambda}^* = \boldsymbol{0} \\ &\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b} = \boldsymbol{0} \end{aligned}$$

i.e.,

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix}\begin{bmatrix} \boldsymbol{x}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} -\boldsymbol{p} \\ \boldsymbol{b} \end{bmatrix} \tag{3.14}$$

see Example 1.3. If $\boldsymbol{H}$ is positive definite and $\boldsymbol{A}$ has full row rank, then the system matrix in (3.14) is nonsingular (explain why?) and the solution $\boldsymbol{x}^*$ from (3.14) is the unique global minimizer of the problem in (3.10). Rather than solving linear system (3.14) of size (n + p), the solution $\boldsymbol{x}^*$ and Lagrange multipliers $\boldsymbol{\lambda}^*$ can be expressed as

$$\boldsymbol{\lambda}^* = -(\boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{A}^T)^{-1}(\boldsymbol{A}\boldsymbol{H}^{-1}\boldsymbol{p} + \boldsymbol{b}) \tag{3.15a}$$

$$\boldsymbol{x}^* = -\boldsymbol{H}^{-1}(\boldsymbol{A}^T\boldsymbol{\lambda}^* + \boldsymbol{p}) \tag{3.15b}$$

■

Example 3.6 *LP with random cost*
Consider an LP problem of the form

$$\begin{aligned}&\text{minimize} && \boldsymbol{c}^T\boldsymbol{x}\\ &\text{subject to:} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \ge \boldsymbol{0}\end{aligned}$$

where $\boldsymbol{c}$ is a *random* vector with mean $\overline{\boldsymbol{c}}$ and covariance $\boldsymbol{\Sigma} = E[(\boldsymbol{c}-\overline{\boldsymbol{c}})(\boldsymbol{c}-\overline{\boldsymbol{c}})^T]$. Consequently, the objective $\boldsymbol{c}^T\boldsymbol{x}$ is *random* with mean $\overline{\boldsymbol{c}}^T\boldsymbol{x}$ (representing expected cost) and variance $\boldsymbol{x}^T\boldsymbol{\Sigma}\,\boldsymbol{x}$ (representing risk). A deterministic way to deal with such LP problems is to examine the convex QP problem

$$\begin{aligned}&\text{minimize} && \overline{\boldsymbol{c}}^T\boldsymbol{x} + \gamma\boldsymbol{x}^T\boldsymbol{\Sigma}\,\boldsymbol{x}\\ &\text{subject to:} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\ \boldsymbol{x} \ge \boldsymbol{0}\end{aligned}$$

instead, where $\gamma > 0$ is a parameter that controls the trade-off between expected cost and variance (risk). ■

3.3 Semidefinite Programming (SDP)

Semidefinite programming (SDP) is a branch of convex programming that has been a subject of intensive research since the early 1990's. The continued interest in SDP has been motivated mainly by two reasons. First, many important classes of optimization problems such as LP and convex QP problems can be viewed as SDP problems, and many CP problems of practical usefulness that are neither LP nor QP problems can also be formulated as SDP problems. Second, several interior-point methods that have proven efficient for LP and convex QP problems have been extended to SDP.

SDP refers to a class of convex problems of the form

$$\begin{aligned}&\text{minimize} && \boldsymbol{c}^T\boldsymbol{x}\\ &\text{subject to:} && F(\boldsymbol{x}) \succeq \boldsymbol{0}\end{aligned} \tag{3.16a-b}$$

where

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{F}_0 + \sum_{i=1}^{p} x_i\boldsymbol{F}_i$$

with $\boldsymbol{F}_i \in S^n$ for $0 \le i \le p$. Note that the positive semidefinite constraint on matrix $\boldsymbol{F}(\boldsymbol{x})$ in (3.16b) is dependent on vector $\boldsymbol{x}$ in an *affine* manner. In the literature, the type of problems described by (3.16) are often referred to as *convex optimization* problems with *linear matrix inequality* (LMI) constraints, and have found many applications in science and engineering. Since the minimization problem in (3.16) is equivalent to a dual SDP problem, the problem itself is often referred to as an SDP problem.

Example 3.7 *LP Problems*

As we have seen earlier, standard-form LP problems can be viewed as a special class of SDP problems where the matrices $\boldsymbol{C}$ and $\boldsymbol{A}_i$ for $1 \le i \le p$ in (3.16) are all diagonal.

The alternative-form LP problem

$$\begin{array}{ll} \text{minimize} & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to:} & \boldsymbol{Ax} \le \boldsymbol{b} \end{array}$$

can be viewed as a linear minimization problem with LMI constraints. This can be demonstrated by expressing matrices $\boldsymbol{F}_i$ for $1 \le i \le p$ in (3.16b) as

$$\boldsymbol{F}_0 = \text{diag}\{\boldsymbol{b}\},\ \boldsymbol{F}_i = -\text{diag}\{\boldsymbol{a}_i\} \quad \text{for } i = 1, 2, \ldots, n$$

where $\boldsymbol{a}_i$ denotes the ith column of $\boldsymbol{A}$. ■

Example 3.8 *Convex QP Problems*

The general convex QP problem

$$\begin{array}{ll} \text{minimize} & \boldsymbol{x}^T\boldsymbol{Hx} + \boldsymbol{p}^T\boldsymbol{x} \\ \text{subject to:} & \boldsymbol{Ax} \le \boldsymbol{b} \end{array} \qquad (3.17\text{a-b})$$

can be formulated as

$$\begin{array}{ll} \text{minimize} & \delta \\ \text{subject to:} & \boldsymbol{x}^T\boldsymbol{Hx} + \boldsymbol{p}^T\boldsymbol{x} \le \delta \\ & \boldsymbol{Ax} \le \boldsymbol{b} \end{array} \qquad (3.18\text{a-c})$$

where δ is an auxiliary scalar variable.

Since $\boldsymbol{H}$ is positive semidefinite, we can find a matrix $\hat{\boldsymbol{H}}$ such that $\boldsymbol{H} = \hat{\boldsymbol{H}}^T\hat{\boldsymbol{H}}$, hence the constraint in (3.18b) can be expressed as

$$\delta - \boldsymbol{p}^T\boldsymbol{x} - (\hat{\boldsymbol{H}}\boldsymbol{x})^T(\hat{\boldsymbol{H}}\boldsymbol{x}) \ge 0 \qquad (3.19)$$

It can be shown that the inequality in (3.19) holds if and only if

$$\boldsymbol{G}(\delta, \boldsymbol{x}) = \begin{bmatrix} \boldsymbol{I}_n & \hat{\boldsymbol{H}}\boldsymbol{x} \\ (\hat{\boldsymbol{H}}\boldsymbol{x})^T & \delta - \boldsymbol{p}^T\boldsymbol{x} \end{bmatrix} \succeq \boldsymbol{0} \qquad (3.20)$$

where $\boldsymbol{I}_n$ is the n by n identity matrix. Note that matrix $\boldsymbol{G}(\delta, \boldsymbol{x})$ is *affine* with respect to variables $\boldsymbol{x}$ and δ. In addition, as shown above the linear constraints in (3.18c) can be expressed as

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{F}_0 + \sum_{i=1}^{n} x_i \boldsymbol{F}_i \succeq \boldsymbol{0}$$

Therefore, by defining an augmented vector

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \delta \\ \boldsymbol{x} \end{bmatrix}$$

the convex QP problem in (3.17) can be reformulated as the SDP problem

$$\begin{aligned} &\text{minimize} \quad \hat{\boldsymbol{c}}^T \hat{\boldsymbol{x}} \\ &\text{subject to:} \ \boldsymbol{E}(\hat{\boldsymbol{x}}) \succeq \boldsymbol{0} \end{aligned}$$

where $\hat{\boldsymbol{c}} = [1\ 0 \ldots 0]^T$ and $\boldsymbol{E}(\hat{\boldsymbol{x}}) = \text{diag}\{\boldsymbol{G}(\delta, \boldsymbol{x}), \boldsymbol{F}(\boldsymbol{x})\}$. ■

Example 3.9 *Convex QP Problems with Quadratic Constraints*
Now let us consider the CP problem

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{p}^T \boldsymbol{x} \\ &\text{subject to:} \ \boldsymbol{x}^T \boldsymbol{Q}_i \boldsymbol{x} + \boldsymbol{q}_i^T \boldsymbol{x} + r_i \le 0 \ \text{ for } \ 1 \le i \le p \end{aligned} \tag{3.21a-b}$$

where $\boldsymbol{H} \succeq \boldsymbol{0}$ and $\boldsymbol{Q}_i \succeq \boldsymbol{0}$ for $1 \le i \le p$. The class of problems represented by (3.21) covers the conventional convex QP problems as a subclass if $\boldsymbol{Q}_i = \boldsymbol{0}$ for all i. Again, by introducing an auxiliary scalar variable δ, the problem in (3.21) can be converted to

$$\begin{aligned} &\text{minimize} \quad \delta \\ &\text{subject to:} \ \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{p}^T \boldsymbol{x} \le \delta \\ &\qquad\qquad\quad \boldsymbol{x}^T \boldsymbol{Q}_i \boldsymbol{x} + \boldsymbol{q}_i^T \boldsymbol{x} + r_i \le 0 \ \text{ for } \ 1 \le i \le p \end{aligned} \tag{3.22a-c}$$

As in the convex QP case, the constraint in (3.22b) is equivalent to the constraint in (3.20) and the constraints in (3.22c) are equivalent to

$$\boldsymbol{F}_i(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{I}_n & \hat{\boldsymbol{Q}}_i \boldsymbol{x} \\ (\hat{\boldsymbol{Q}}_i \boldsymbol{x})^T & -\boldsymbol{q}_i^T \boldsymbol{x} - r_i \end{bmatrix} \succeq \boldsymbol{0} \ \text{ for } \ 1 \le i \le p$$

where $\hat{\boldsymbol{Q}}_i$ is related to $\boldsymbol{Q}_i$ by the equation $\boldsymbol{Q}_i = \hat{\boldsymbol{Q}}_i^T \hat{\boldsymbol{Q}}$. Consequently, the problem in (3.21) can be formulated as

$$\begin{aligned} &\text{minimize} \quad \hat{\boldsymbol{c}}^T \hat{\boldsymbol{x}} \\ &\text{subject to:} \ \boldsymbol{E}(\hat{\boldsymbol{x}}) \succeq \boldsymbol{0} \end{aligned} \tag{3.23a-b}$$

where

$$\boldsymbol{E}(\hat{\boldsymbol{x}}) = \text{diag}\{G(\delta, \boldsymbol{x}), \boldsymbol{F}_1(\boldsymbol{x}), \boldsymbol{F}_2(\boldsymbol{x}), \ldots, \boldsymbol{F}_p(\boldsymbol{x})\}$$

■

Example 3.10 *Minimizing largest eigenvalue of an affine matrix*

An affine matrix is a symmetric matrix of the form $\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{A}_0 + x_1 \boldsymbol{A}_1 + \cdots + x_p \boldsymbol{A}_p$ where $\boldsymbol{A}_i \in S^n$

for i = 0, 1, …, p are known symmetric matrices and $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_p \end{bmatrix}^T$ is an unknown parameter vector to be determined by solving

$$\text{minimize} \quad \lambda_{\max}(\boldsymbol{A}(\boldsymbol{x}))$$

where $\lambda_{\max}(\boldsymbol{A}(\boldsymbol{x}))$ denotes the largest eigenvalue of $\boldsymbol{A}(\boldsymbol{x})$. From linear algebra we know that the smallest scalar t such that $t\boldsymbol{I} - \boldsymbol{A}$ becomes positive semidefinite is $t = \lambda_{\max}(\boldsymbol{A})$. Based on this, the above problem can be formulated as

$$\begin{aligned} &\text{minimize} \quad t \\ &\text{subject to: } t\boldsymbol{I} - (\boldsymbol{A}_0 + x_1\boldsymbol{A}_1 + \cdots + x_p\boldsymbol{A}_p) \succeq \boldsymbol{0} \end{aligned}$$

where both t and $\boldsymbol{x}$ are treated as unknown variables. ■

3.4 Second-Order Cone Programming (SOCP)

Definition 3.1 A second-order cone of dimension n is defined as

$$\mathcal{K} = \left\{ \begin{bmatrix} t \\ \boldsymbol{u} \end{bmatrix} : t \in R, \boldsymbol{u} \in R^{n-1} \text{ for } \|\boldsymbol{u}\|_2 \le t \right\} \tag{3.24}$$

A second-order cone is also called *quadratic* or *Lorentz cone*.
For n = 1, the second-order cone degenerates into a ray on the t axis starting from t = 0, as shown in Fig. 3.1a. The second-order cones for n = 2 and 3 are depicted in Fig. 3.1b and c, respectively.

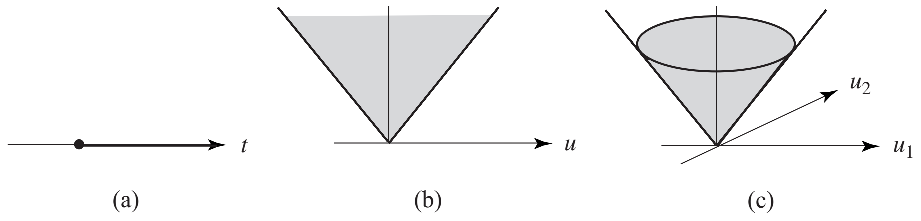


Figure 3.1 Second-order cones of dimension (a) n = 1, (b) n = 2, and (c) n = 3.

Note that the second-order cone $\mathcal{K}$ is a convex set in R^n because for any two points in $\mathcal{K}$, $[t_1 \; \boldsymbol{u}_1^T]^T$ and $[t_2 \; \boldsymbol{u}_2^T]^T$, and $\lambda \in [0, 1]$, we have

$$\lambda \begin{bmatrix} t_1 \\ \boldsymbol{u}_1 \end{bmatrix} + (1-\lambda) \begin{bmatrix} t_2 \\ \boldsymbol{u}_2 \end{bmatrix} = \begin{bmatrix} \lambda t_1 + (1-\lambda) t_2 \\ \lambda \boldsymbol{u}_1 + (1-\lambda)\boldsymbol{u}_2 \end{bmatrix}$$

where

$$\|\lambda \boldsymbol{u}_1 + (1-\lambda)\boldsymbol{u}_2\|_2 \le \lambda \|\boldsymbol{u}_1\|_2 + (1-\lambda)\|\boldsymbol{u}_2\|_2 \le \lambda t_1 + (1-\lambda) t_2$$

A general second-order cone-programming (SOCP) problem assumes the form

$$\begin{aligned}&\text{minimize} \quad \boldsymbol{c}^T\boldsymbol{x}\\ &\text{subject to: } \|\boldsymbol{A}_i^T\boldsymbol{x}+\boldsymbol{c}_i\|_2 \le \boldsymbol{b}_i^T\boldsymbol{x}+d_i \ \text{ for } 1\le i\le q\end{aligned} \tag{3.25a-b}$$

The constraints in (3.25b) are equivalent to

$$\text{vector } \begin{bmatrix}\boldsymbol{b}_i^T\boldsymbol{x}+d_i\\ \boldsymbol{A}_i^T\boldsymbol{x}+\boldsymbol{c}_i\end{bmatrix} \text{ belongs to a second-order cone for } i = 1, 2, \dots q.$$

Relations between LP, QP, SDP and SOCP

The class of SOCP problems is large enough to include both LP and convex QP problems.

- If $\boldsymbol{A}_i = \boldsymbol{0}$ and $\boldsymbol{c}_i = \boldsymbol{0}$ for $i = 1, 2, \dots, q$, then the problem in (3.91) becomes

$$\begin{aligned}&\text{minimize} \quad \boldsymbol{b}^T\boldsymbol{x}\\ &\text{subject to: } \boldsymbol{b}_i^T\boldsymbol{x}+d_i \ge 0 \ \text{ for } 1\le i\le q\end{aligned}$$

which is obviously an LP problem.

- Now consider the convex QP problem

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}+2\boldsymbol{x}^T\boldsymbol{p}\\ &\text{subject to: } \boldsymbol{A}\boldsymbol{x}\le\boldsymbol{b}\end{aligned} \tag{3.26a-b}$$

where $\boldsymbol{H}$ is positive definite. If we write matrix $\boldsymbol{H}$ as $\boldsymbol{H} = \boldsymbol{H}^{T/2}\boldsymbol{H}^{1/2}$ and let $\tilde{\boldsymbol{p}} = \boldsymbol{H}^{-T/2}\boldsymbol{p}$, then the objective function in (3.26a) can be expressed as

$$f(\boldsymbol{x}) = \|\boldsymbol{H}^{1/2}\boldsymbol{x}+\tilde{\boldsymbol{p}}\|_2^2 - \boldsymbol{p}^T\boldsymbol{H}^{-1}\boldsymbol{p}$$

Since the term $\boldsymbol{p}^T\boldsymbol{H}^{-1}\boldsymbol{p}$ is a constant, minimizing $f(\boldsymbol{x})$ is equivalent to minimizing $\|\boldsymbol{H}^{1/2}\boldsymbol{x}+\tilde{\boldsymbol{p}}\|_2$ and thus the problem at hand can be converted to

$$\begin{aligned}&\text{minimize} \quad \delta\\ &\text{subject to: } \|\boldsymbol{H}^{1/2}\boldsymbol{x}+\tilde{\boldsymbol{p}}\|_2 \le \delta\\ &\qquad\qquad\quad \boldsymbol{A}\boldsymbol{x}\le\boldsymbol{b}\end{aligned}$$

where δ is an upper bound for $\|\boldsymbol{H}^{1/2}\boldsymbol{x}+\tilde{\boldsymbol{p}}\|_2$ that can be treated as an auxiliary variable of the problem. By defining

$$\tilde{\boldsymbol{x}} = \begin{bmatrix}\delta\\ \boldsymbol{x}\end{bmatrix},\ \tilde{\boldsymbol{c}} = \begin{bmatrix}1\\ \boldsymbol{0}\end{bmatrix},\ \tilde{\boldsymbol{H}} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{H}^{1/2}\end{bmatrix},\ \tilde{\boldsymbol{A}} = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{A}\end{bmatrix}$$

the problem becomes

$$\begin{aligned}&\text{minimize} \quad \tilde{\boldsymbol{c}}^T\tilde{\boldsymbol{x}}\\ &\text{subject to: } \|\tilde{\boldsymbol{H}}\tilde{\boldsymbol{x}}+\tilde{\boldsymbol{p}}\|_2 \le \tilde{\boldsymbol{c}}^T\tilde{\boldsymbol{x}}\\ &\qquad\qquad\quad \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}\le\boldsymbol{b}\end{aligned}$$

which is an SOCP problem.

• On the other hand, it can be shown that every SOCP problem can be formulated as an SDP problem. To see this, note that the constraint $\|\boldsymbol{u}\|_2 \le t$ implies that

$$\begin{bmatrix} t\boldsymbol{I} & \boldsymbol{u} \\ \boldsymbol{u}^T & t \end{bmatrix} \succeq \mathbf{0}$$

In other words, a second-order cone can be embedded into a cone of positive semidefinite matrices, and the SOCP problem in (3.25) can be formulated as

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{c}^T\boldsymbol{x} \\ &\text{subject to:} \quad \begin{bmatrix} (\boldsymbol{b}_i^T\boldsymbol{x}+d_i)\boldsymbol{I} & \boldsymbol{A}_i^T\boldsymbol{x}+\boldsymbol{c}_i \\ (\boldsymbol{A}_i^T\boldsymbol{x}+\boldsymbol{c}_i)^T & \boldsymbol{b}_i^T\boldsymbol{x}+d_i \end{bmatrix} \succeq \mathbf{0} \end{aligned}$$

which is an SDP problem.

The above analysis has demonstrated that the branch of nonlinear programming known as CP can be subdivided into a series of nested branches of optimization, namely, SDP, SOCP, convex QP, and LP as illustrated in Fig. 3.2.

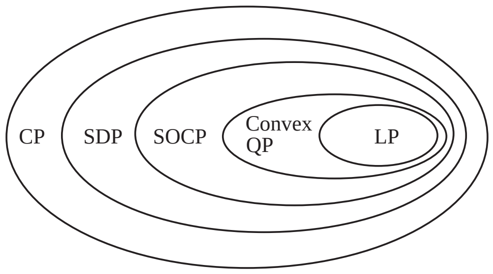


Figure 3.2 Relations among LP, convex QP, SOCP, SDP, and CP problems.

Example 3.11 *QP with quadratic constraints*

A general QP problem with quadratic constrains can be expressed as

$$\begin{aligned} &\text{minimize} \quad \boldsymbol{x}^T\boldsymbol{H}_0\boldsymbol{x} + 2\boldsymbol{p}_0^T\boldsymbol{x} \\ &\text{subject to:} \quad \boldsymbol{x}^T\boldsymbol{H}_i\boldsymbol{x} + 2\boldsymbol{p}_i^T\boldsymbol{x} + r_i \le 0 \quad \text{for } 1 \le i \le q \end{aligned} \tag{3.27a-b}$$

where $\boldsymbol{H}_i$ for i = 0, 1, …, q are assumed to be positive definite matrices. Using the matrix decomposition $\boldsymbol{H}_i = \boldsymbol{H}_i^{T/2}\boldsymbol{H}_i^{1/2}$, the problem in (3.27) can be expressed as

$$\begin{aligned} &\text{minimize} \quad \|\boldsymbol{H}_0^{1/2}\boldsymbol{x} + \tilde{\boldsymbol{p}}_0\|_2^2 - \boldsymbol{p}_0^T\boldsymbol{H}_0^{-1}\boldsymbol{p}_0 \\ &\text{subject to:} \quad \|\boldsymbol{H}_i^{1/2}\boldsymbol{x} + \tilde{\boldsymbol{p}}_i\|_2^2 - \boldsymbol{p}_i^T\boldsymbol{H}_i^{-1}\boldsymbol{p}_i + r_i \le 0 \quad \text{for } 1 \le i \le q \end{aligned}$$

where $\tilde{\boldsymbol{p}}_i = \boldsymbol{H}_i^{-T/2}\boldsymbol{p}_i$ for i = 0, 1, …, q. Obviously, the above problem is equivalent to the SOCP problem

$$\begin{array}{ll}\text{minimize} & \delta \\ \text{subject to:} & \|\boldsymbol{H}_0^{1/2}\boldsymbol{x} + \tilde{\boldsymbol{p}}_0\|_2 \le \delta \\ & \|\boldsymbol{H}_i^{1/2}\boldsymbol{x} + \tilde{\boldsymbol{p}}_i\|_2 \le (\boldsymbol{p}_i^T \boldsymbol{H}_i^{-1}\boldsymbol{p}_i - r_i)^{1/2} \quad \text{for } 1 \le i \le q\end{array}$$

■

Example 3.12 *Minimization of sum of L*$_2$ *norms*

Unconstrained minimization problems of the form

$$\text{minimize} \ \sum_{i=1}^{N} \|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{c}_i\|_2$$

occur in a number of applications. By introducing an upper bound for each L_2-norm term in the objective function, the problem can be converted to

$$\begin{array}{ll}\text{minimize} & \sum_{i=1}^{N} \delta_i \\ \text{subject to:} & \|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{c}_i\|_2 \le \delta_i \ \text{ for } 1 \le i \le N\end{array} \qquad (3.28\text{a-b})$$

If we define

$$\tilde{\boldsymbol{x}} = \begin{bmatrix}\delta_1 \\ \vdots \\ \delta_N \\ \boldsymbol{x}\end{bmatrix} \quad \tilde{\boldsymbol{c}} = \begin{bmatrix}1 \\ \vdots \\ 1 \\ \boldsymbol{0}\end{bmatrix}, \ \tilde{\boldsymbol{A}}_i = \begin{bmatrix}\boldsymbol{0} & \boldsymbol{A}_i\end{bmatrix}, \ \tilde{\boldsymbol{b}}_i = \begin{bmatrix}0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0\end{bmatrix} \begin{matrix} \\ \\ \\ \leftarrow i\text{th element} \\ \\ \\ \\ \end{matrix}$$

then (3.28) becomes

$$\begin{array}{ll}\text{minimize} & \tilde{\boldsymbol{c}}^T\tilde{\boldsymbol{x}} \\ \text{subject to:} & \|\tilde{\boldsymbol{A}}_i^T\tilde{\boldsymbol{x}} + \boldsymbol{c}_i\|_2 \le \tilde{\boldsymbol{b}}_i^T\tilde{\boldsymbol{x}} \quad \text{for } 1 \le i \le N\end{array}$$

which is an SOCP problem.

Another unconstrained problem related to the problem in (3.28) is the *minimax problem*

$$\underset{\boldsymbol{x}}{\text{minimize}} \ \underset{1\le i\le N}{\text{maximize}} \ \|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{c}_i\|_2$$

which can be re-formulated as the SOCP problem

$$\begin{array}{ll}\text{minimize} & \delta \\ \text{subject to:} & \|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{c}_i\|_2 \le \delta \ \text{ for } 1 \le i \le N\end{array}$$

■

Example 3.13 *Complex L*$_1$*-norm approximation*

An interesting special case of the sum-of-norms problem is the complex L_1 norm approximation problem whereby a complex-valued approximate solution for the linear equation $\boldsymbol{Ax} = \boldsymbol{b}$ is required where $\boldsymbol{A}$ and $\boldsymbol{b}$ are complex-valued such that $\boldsymbol{x}$ solves the unconstrained problem

$$\text{minimize} \quad \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{c}\|_1$$

where $\boldsymbol{A} \in C^{m\times n}, \boldsymbol{c} \in C^{m\times 1}, \boldsymbol{x} \in C^{n\times 1}$ and the L_1 norm of $\boldsymbol{x}$ is defined as $\|\boldsymbol{x}\|_1 = \sum_{k=1}^{n} |x_i|$. If we let $\boldsymbol{A} = [\boldsymbol{a}_1\ \boldsymbol{a}_2 \cdots \boldsymbol{a}_m]^T$ and $\boldsymbol{c} = [c_1\ c_2 \cdots c_m]^T$ where $\boldsymbol{a}_k = \boldsymbol{a}_{kr} + j\boldsymbol{a}_{ki}, c_k = c_{kr} + jc_{ki}$ and $j = \sqrt{-1}$, then we have

$$\begin{aligned}
\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{c}\|_1 &= \sum_{k=1}^{m} |\boldsymbol{a}_k^T\boldsymbol{x} - c_k| \\
&= \sum_{k=1}^{m} \left[\left(\boldsymbol{a}_{kr}^T\boldsymbol{x}_r - \boldsymbol{a}_{ki}^T\boldsymbol{x}_i - c_{kr}\right)^2 + \left(\boldsymbol{a}_{kr}^T\boldsymbol{x}_i + \boldsymbol{a}_{ki}^T\boldsymbol{x}_r - c_{ki}\right)^2 \right]^{1/2} \\
&= \sum_{k=1}^{m} \left\| \begin{bmatrix} \boldsymbol{a}_{kr}^T & -\boldsymbol{a}_{ki}^T \\ \boldsymbol{a}_{ki}^T & \boldsymbol{a}_{kr}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_r \\ \boldsymbol{x}_i \end{bmatrix} - \begin{bmatrix} c_{kr} \\ c_{ki} \end{bmatrix} \right\|_2 \triangleq \sum_{k=1}^{m} \|\boldsymbol{A}_k\hat{\boldsymbol{x}} - \boldsymbol{c}_k\|_2
\end{aligned}$$

Hence the problem under consideration can be converted to minimize

$$\begin{aligned}
&\text{minimize} \quad \sum_{k=1}^{m} \delta_k \\
&\text{subject to: } \|\boldsymbol{A}_k\hat{\boldsymbol{x}} + \boldsymbol{c}_k\|_2 \le \delta_k \ \text{ for } 1 \le k \le m
\end{aligned} \tag{3.29a-b}$$

By letting

$$\tilde{\boldsymbol{x}} = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_m \\ \hat{\boldsymbol{x}} \end{bmatrix}, \ \tilde{\boldsymbol{c}}_0 = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \boldsymbol{0} \end{bmatrix}, \ \tilde{\boldsymbol{A}}_k = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{A}_k \end{bmatrix}, \ \tilde{\boldsymbol{b}}_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \leftarrow k\text{th element}$$

the problem in (3.29) becomes

$$\begin{aligned}
&\text{minimize} \quad \tilde{\boldsymbol{c}}_0^T \tilde{\boldsymbol{x}} \\
&\text{subject to: } \|\tilde{\boldsymbol{A}}_k\tilde{\boldsymbol{x}} + \boldsymbol{c}_k\|_2 \le \tilde{\boldsymbol{b}}_k^T \tilde{\boldsymbol{x}} \ \text{ for } 1 \le k \le m
\end{aligned}$$

which is obviously an SOCP problem. ■

Example 3.14 *Linear fractional problem*

The linear fractional problem can be described as

$$\begin{aligned}
&\text{minimize} \quad \sum_{i=1}^{p} \frac{1}{\boldsymbol{a}_i^T\boldsymbol{x} + c_i} \\
&\text{subject to: } \boldsymbol{a}_i^T\boldsymbol{x} + c_i > 0 \ \text{ for } 1 \le i \le p \\
&\qquad\qquad\quad \boldsymbol{b}_j^T\boldsymbol{x} + d_j \ge 0 \ \text{ for } 1 \le j \le q
\end{aligned} \tag{3.30a-c}$$

It can be readily verified that subject to the constraints in (3.30b), each term in the objective function is convex and hence the objective function itself is also convex. It, therefore, follows that the problem in (3.30) is a CP problem. By introducing the auxiliary constraints

$$\frac{1}{\boldsymbol{a}_i^T \boldsymbol{x} + c_i} \le \delta_i$$

i.e., $$\delta_i(\boldsymbol{a}_i^T \boldsymbol{x} + c_i) \ge 1$$

and $$\delta_i \ge 0$$

for $i = 1, 2, \ldots, p$, the problem in (3.30) can be expressed as

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{p} \delta_i \\ \text{subject to:} \quad & \delta_i(\boldsymbol{a}_i^T \boldsymbol{x} + c_i) \ge 1 \;\; \text{for} \;\; 1 \le i \le p \\ & \delta_i \ge 0 \\ & \boldsymbol{b}_j^T \boldsymbol{x} + d_j \ge 0 \;\; \text{for} \;\; 1 \le j \le q \end{aligned} \tag{3.31a-d}$$

Furthermore, we note that scalars *u, v,* and *w* satisfy $w^2 \le uv, u \ge 0, v \ge 0$ if and only if

$$\left\| \begin{bmatrix} 2w \\ u - v \end{bmatrix} \right\|_2 \le u + v$$

hence the constraints in (3.31b) and (3.31c) can be written as

$$\left\| \begin{bmatrix} 2 \\ \boldsymbol{a}_i^T \boldsymbol{x} + c_i - \delta_i \end{bmatrix} \right\|_2 \le \boldsymbol{a}_i^T \boldsymbol{x} + c_i + \delta_i \;\; \text{for} \;\; 1 \le i \le p$$

Hence the problem in (3.31) can be formulated as

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{p} \delta_i \\ \text{subject to:} \quad & \left\| \begin{bmatrix} 2 \\ \boldsymbol{a}_i^T \boldsymbol{x} + c_i - \delta_i \end{bmatrix} \right\|_2 \le \boldsymbol{a}_i^T \boldsymbol{x} + c_i + \delta_i \;\; \text{for} \;\; 1 \le i \le p \\ & \boldsymbol{b}_j^T \boldsymbol{x} + d_j \ge 0 \;\; \text{for} \;\; 1 \le j \le q \end{aligned}$$

which is an SOCP problem. ■

3.5 Software

In this section we introduce a popular CP solver known as **CVX**. The material of this section is largely based on the User Guide version 1.22 of the software written by M. Grant and S. Boyd.

3.5.1 What is CVX?

Designed by Michael Grant and Stephen Boyd, with input from Yinyu Ye, **CVX** is a modeling system for disciplined convex programming that are convex optimization problems described by a limited set of construction rules. **CVX** solves standard problems such as LP, QP, SOCP, and SDP problems. Compared to directly using a solver such as SeDuMi for these problems, **CVX** greatly simplifies the task of specifying the problem. **CVX** also solves more complex convex optimization problems, including many involving nonsmooth functions, such as l_1 norm.

3.5.2 Examples

Example 3.15

We first consider the most basic convex optimization problem, least-squares. In a least-squares problem, we seek $\boldsymbol{x} \in R^n$ that minimizes $\|\boldsymbol{Ax} - \boldsymbol{b}\|_2$, where $\boldsymbol{A} \in R^{m\times n}$ is skinny and full rank (i.e., $m \ge n$ and rank($\boldsymbol{A}$) = n). Let us create some test problem data for m, n, $\boldsymbol{A}$, and $\boldsymbol{b}$ in Matlab:

```
m = 16; n = 8;
A = randn(m,n);
b = randn(m,1);
```

(Here we chose small values of m and n to keep the output readable.) Then the least squares solution $\boldsymbol{x} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$ is easily computed using the backslash operator:

```
x_ls = A\b;
```

Using **CVX**, the same problem can be solved as follows:

```
cvx_begin
    variable x(n);
    minimize(norm(A*x-b));
cvx_end
```

When MATLAB reaches command **cvx_end**, the least-squares problem is solved, and the Matlab variable **x** is overwritten with the solution of the least-squares problem, i.e., $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$. Now **x** is an ordinary length-n numerical vector, identical to what would be obtained in the traditional approach, at least to within the accuracy of the solver. In addition, two additional Matlab variables are created:

- **cvx_optval**, which contains the value of the objective function; i.e., $\|\boldsymbol{Ax} - \boldsymbol{b}\|_2$;
- **cvx_status**, which contains a string describing the status of the calculation. In this case, **cvx_status** would contain the string Solved. See Appendix C of the user guide for a list of the possible values of **cvx_status** and their meaning.
- **cvx_slvtol**: the tolerance level achieved by the solver.
- **cvx_slvitr**: the number of iterations taken by the solver.

Example 3.16

Suppose we wish to add some simple upper and lower bounds to the least-squares problem above: i.e., we wish to solve

$$\begin{aligned} &\text{minimize} \quad \|\boldsymbol{Ax} - \boldsymbol{b}\|_2 \\ &\text{subject to:} \quad \boldsymbol{l} \le \boldsymbol{x} \le \boldsymbol{u} \end{aligned}$$

where $\boldsymbol{l}$ and $\boldsymbol{u}$ are given data, vectors with the same dimension as the variable $\boldsymbol{x}$. We can no longer use the simple backslash notation to solve this problem, but it can be transformed into a

QP, which can be solved without difficulty if you have some form of QP software available.

Let us provide some numeric values for $\boldsymbol{l}$ and $\boldsymbol{u}$:

```
bnds = randn(n,2);
l = min(bnds,[],2);
u = max(bnds,[],2);
```

Then if you have MATLAB Optimization Toolbox, you can use function quadprog to solve the problem as follows:

```
x_qp = quadprog(2*A'*A,-2*A'*b,[],[],[],[],l,u);
```

This actually minimizes the square of the norm, which is the same as minimizing the norm itself. In contrast, the **CVX** specification is given by

```
cvx_begin
   variable x(n);
   minimize(norm(A*x-b));
   subject to
   l <= x <= u;
cvx_end
```

Example 3.17

CVX supports constraints more complex than simple bounds as seen in Example 3.16. For example, let us define new matrices $\boldsymbol{C}$ and $\boldsymbol{d}$ in MATLAB as follows,

```
p = 4;
C = randn(p,n);
d = randn(p,1);
```

Now let us add an equality constraint and a nonlinear inequality constraint to the original least-squares problem:

```
cvx_begin
   variable x(n);
   minimize(norm(A*x-b));
   subject to
   C*x == d;
   norm(x,Inf) <= 1;
cvx_end
```

Example 3.18

For our final example in this section, let us show how traditional Matlab code and **cvx** specifications can be mixed to form and solve multiple optimization problems. The following code solves the problem of minimizing $\| \boldsymbol{Ax}-\boldsymbol{b} \|_2 + \gamma \| \boldsymbol{x} \|_1$, for a logarithmically spaced vector of (positive) values of γ. This gives us points on the optimal trade-off curve between $\| \boldsymbol{Ax}-\boldsymbol{b} \|_2$ and $\| \boldsymbol{x} \|_1$. An example of this curve is given in Fig. 3.3.

```
gamma = logspace(-2,2,20);
l2norm = zeros(size(gamma));
l1norm = zeros(size(gamma));
fprintf(1,'gamma norm(x,1) norm(A*x-b)\n');
fprintf(1,'---------------------------------------\n');
for k = 1:length(gamma),
   fprintf(1,'%8.4e',gamma(k) );
   cvx_begin quiet
      variable x(n);
      minimize(norm(A*x-b)+gamma(k)*norm(x,1));
   cvx_end
   l1norm(k) = norm(x,1);
   l2norm(k) = norm(A*x-b);
   fprintf(1,'%8.4e %8.4e\n',l1norm(k),l2norm(k));
end
plot(l1norm,l2norm);
xlabel('norm(x,1)');
ylabel('norm(A*x-b)');
grid
```

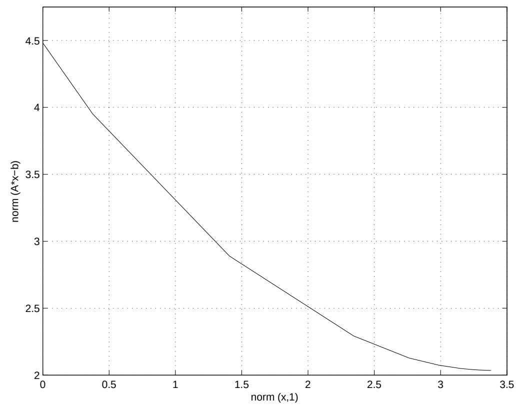


Figure 3.3

3.5.3 Basic Rules

(1) All **CVX** models must be preceded by command **cvx_begin** and terminated with commend **cvx_end**. Command **cvx_begin** accepts several variants. For example, **cvx_begin quiet** prevents the model from producing screen output while it is being solved.

(2) Data Types for Variables

All variables must be declared using command variable (or variables). Variables can be real or complex; and scalar, vector, matrix, or *n*-dimensional array. Variables can be matrices with structure, like symmetry or bandedness. The variable structure is given by including descriptive keywords following the name and size of the variable. Here are some examples of **CVX** variable:

```
        variable w(50) complex;
        variable X(20,10);
        variable Y(50,50) symmetric;
        variable Z(100,100) hermitian toeplitz;
```

- **Structure Keywords**

Version 1.22 of **`CVX`** supports the following structure keywords:

`banded(lb,ub) complex diagonal hankel hermitian lower_bidiagonal lower_hessenberg lower_triangular scaled_identity skew_symmetric symmetric toeplitz tridiagonal upper_bidiagonal upper_hankel upper_hessenberg upper_triangular`

(3) Objective functions

Declaring an objective function requires the use of the minimize or maximize function, as appropriate. The objective function in a call to minimize must be convex; the objective function in a call to maximize must be concave. At most one objective function may be declared in a given **`CVX`** specification, and the objective function must have a scalar value.

If no objective function is specified, the problem is interpreted as a *feasibility* problem, which is the same as performing a minimization with the objective function set to zero. In this case, **`cvx_optval`** is either **`0`**, if a feasible point is found, or **`+Inf`**, if the constraints are not feasible.

(4) Constraints

Version 1.22 of **`CVX`** supports the following constraint types:

• Equality **==** constraints, where both the left- and right-hand sides are affine functions of the optimization variables.

• Less-than **<=** inequality constraints, where the left-hand expression is convex, and the right-hand expression is concave.

• Greater-than **>=** constraints, where the left-hand expression is concave, and the right-hand expression is convex.

These equality and inequality operators work for arrays. When both sides of the constraint are arrays of the same size, the constraint is imposed elementwise. For example, if **`a`** and **`b`** are $m \times n$ matrices, then **`a <= b`** is interpreted by **`CVX`** as mn (scalar) inequalities, i.e., each entry of **`a`** must be less than or equal to the corresponding entry of **`b`**. **`CVX`** also handles cases where one side is a scalar and the other is an array. This is interpreted as a constraint for each element of the array, with the (same) scalar appearing on the other side. As an example, if **`a`** is an $m \times n$ matrix, then **`a >= 0`** is interpreted as mn inequalities: each element of the matrix must be nonnegative.

(5) Functions

The base **`CVX`** function library includes a variety of convex, concave, and affine functions which accept **`CVX`** variables or expressions as arguments. Many are common Matlab functions such as **`sum, trace, diag, sqrt, max,`** and **`min`**, re-implemented as needed to support **`CVX`**; others are new functions not found in MATLAB. A complete list of the functions in the base library can be found in § B of the user guide. It's also possible to add your own new functions; see §5 of the user guide.

(6) Sets

- **`CVX`** supports the definition and use of convex sets. The base library includes the cone of positive semidefinite $n \times n$ matrices, the second-order or Lorentz cone, and various norm balls. A complete list of sets supplied in the base library is given in § B of the user guide.

• Since MATLAB does not have a set membership operator, **CVX** adopts a slightly different syntax to require that an expression is in a set. To represent a set, a function is used to return an unnamed variable that is required to be in the set. Consider, for example, $\boldsymbol{S}_{+}^{n}$, the cone of symmetric positive semidefinite $n \times n$ matrices. In **CVX**, this is represented by function semidefinite(n), which returns an unnamed new variable, that is constrained to be positive semidefinite. To require that the matrix expression **X** be symmetric positive semidefinite, the syntax **X == semidefinite(n)** is used. The literal meaning of this is that **X** is constrained to be equal to some unnamed variable, which is required to be an $n \times n$ symmetric positive semidefinite matrix. This is equivalent to saying that **X** must be symmetric positive semidefinite.

As an example, consider the constraint that a (matrix) variable **X** is a correlation matrix, i.e., it is symmetric, has unit diagonal elements, and is positive semidefinite.

• In **CVX** we can declare such a variable and impose such constraints using

```
variable X(n,n) symmetric;
X == semidefinite(n);
diag(X) == ones(n,1);
```

The second line here imposes the constraint that **X** be positive semidefinite (You can read '**==**' here as 'is', so the second line can be read as '**X** is positive semidefinite'). The left-hand side of the third line is a vector containing the diagonal elements of **X**, whose elements we require to be equal to one. Incidentally, **CVX** allows us to simplify the third line to

```
diag(X) == 1;
```

because **CVX** follows the Matlab convention of handling array/scalar comparisons by comparing each element of the array independently with the scalar.

• Sets can be combined in affine expressions, and we can constrain an affine expression to be in a convex set. For example, we can impose constraints of the form

```
A*X*A'-X == B*semidefinite(n)*B';
```

where **X** is an $n \times n$ symmetric variable matrix, and $\boldsymbol{A}$ and $\boldsymbol{B}$ are $n \times n$ constant matrices. This constraint requires that $\boldsymbol{AXA}^{T} - \boldsymbol{X} = \boldsymbol{BYB}^{T}$, for some $Y \in \boldsymbol{S}_{+}^{n}$.

3.5.4 More Examples

Example 3.19 Standard-form LP

```
function x = lps_cvx(c,A,b)
n = length(c);
cvx_begin quiet
  variable x(n);
  minimize(c'*x);
  subject to
  A*x == b;
  x >= 0;
```

```
cvx_end
```

Example 3.20 Alternative-form LP

```
function x = lpa_cvx(c,A,b)
n = length(c);
cvx_begin quiet
  variable x(n);
  minimize(c'*x);
  subject to
  A*x >= b;
cvx_end
```

Example 3.21 Convex QP

```
function x = qp_cvx(H,p,A,b,E,d)
n = length(p);
cvx_begin quiet
  variable x(n);
  minimize(0.5*x'*H*x+x'*p);
  subject to
  A*x == b;
  E*x >= d;
cvx_end
```

Example 3.22 SDP

```
% Program: sdp_cvx.m
% To solve semidefinite programming (SDP) problem
%           minimize   c'*x
%           subject to: F0 + x(1)*F1 + ... + x(p)*Fp >= 0
% using CVX, where ">= 0" means being positive semidefinite,
% and x = [x1 x2 ... xp]'.
% Input:
% c: constant vector of length p defining objective function.
% F = [F0 F1 ... Fp] with each Fi a symmetric square matrix.
% Output:
% x: solution of the SDP problem.
% Example: solve the SDP problem involved in Example 14.1
% from PO, where
% c = [0 0 0 -1]';
% F = [F0 F1 F2 F3 F4]
%  = [-2.0 0.5 0.6 0 -1 0 0 0 -1 0  0  0 -1  0  0;
%     0.5 -2.0 -0.4 -1  0 0 0 0 0 0 0 -1 0 -1 0;
%    0.6 -0.4 -3.0 0 0  0 -1 0 0 0 -1 0 0 0 -1];
% x = sdp_cvx(c,F);
function x = sdp_cvx(c,F)
p = length(c);
[n,m] = size(F);
Fx = F(:,1:n);
cvx_begin quiet
  variable x(p);
  minimize(c'*x);
  subject to
  for i = 1:p,
      Fx = Fx + x(i)*F(:,(i*n+1):((i+1)*n));
  end
  Fx == semidefinite(n);
cvx_end
```

Example 3.23 SOCP

```
% Program: socp_cvx.m
% To solve second-order cone programming (SOCP) problem
%        minimize b'*x
%        subject to: ||Ai'*x + ci|| <= bi'*x + di
%                  for i = 1, ..., q
% using CVX, where x = [x1 x2 ... xm]' and Ai is a matrix of
% size m by ni. Thus there are a total of q 2nd-order (Lorentz)
% cones, each with size ni.
% Input:
% A = [A1 A2 ... Aq]
% B = [b1 b2 ... bq]
% b: constant vector of length m defining objective function
% c = [c1; c2; ...; cq]
% d = [d1 d2 ... dq]
% k = [n1 n2 ... nq]
% Output:
% x: solution of the SOCP problem.
% Example: Solve the SOCP problem in Example 14.5 from PO, where
% b = [1 0 0 0 0]';
% A = [0 0 0 0 0 0;
%     -1 0 0.5 0 0 0;
%      0 1 0 1 0 0;
%      1 0 0 0 -0.7071 -0.3536;
%      0 -1 0 0 -0.7071 0.3536];
% z = zeros(5,1);B = [b z z];
% c = [0 0 -0.5 0 4.2426 -0.7071]';
% d = [0 1 1];
% k = [2 2 2];
% x = socp_cvx(A,B,b,c,d,k);
function x = socp_cvx(A,B,b,c,d,k)
m = length(b);
q = length(d);
t = 0;
cvx_begin quiet
  variable x(m);
  minimize(b'*x);
  subject to
  for i = 1:q,
      Ai = A(:,(t+1):(t+k(i)));
      bi = B(:,i);
      ci = c((t+1):(t+k(i)));
      di = d(i);
      norm(Ai'*x+ci) <= bi'*x + di;
      t = t + k(i);
  end
cvx_end
```

Problems

3.1 Give an explicit solution of the LP problem

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x}\\ &\text{subject to: } \textstyle\sum_{i=1}^{n} x_i = \alpha \ \text{ and } \ 0 \le x_i \le 1 \text{ for } i = 1, 2, \ldots .n\end{aligned}$$

without using computer code, where α is an integer between 1 and n.

3.2 (a) Find Chebyshev centre $\boldsymbol{x}_c$ for the polygon $\mathcal{P}$ defined by the linear constraints

$$\mathcal{P}: \begin{cases} -x_2 \le 0 \\ -x_1 + x_2 \le 1 \\ 3x_1 - x_2 \le 1.5 \\ x_1 + x_2 \le 3 \end{cases}$$

and the radius r of the largest disk $\mathcal{D}$ inscribed in $\mathcal{P}$.

(b) Illustrate the results obtained in part (a) by displaying $\mathcal{P}$, $\mathcal{D}$, and $\boldsymbol{x}_c$ in a figure.

3.3 It is well known that redundant linear equality constraints can be identified and removed using SVD. But how to identify remove redundant inequality constraints from a given constrained problem? To be precise, consider the polygon defined by the following set of linear inequality constraints:

$$\begin{aligned} 2x_1 + x_2 - 2 &\le 0 \\ -x_2 &\le 0 \\ x_1 - x_2 - 1 &\le 0 \\ x_1 + x_2 - 1 &\le 0 \end{aligned}$$

Develop a numerical method (without using geometry) to verify if the first constraint is redundant and demonstrate the validity of the method by numerical calculations. (Hint: convert the problem at hand into an LP problem.)

3.4 Formulate the following problems as LP problems.

(a) Minimize $\|\boldsymbol{Ax} - \boldsymbol{b}\|_\infty$.

(b) Minimize $\|\boldsymbol{Ax} - \boldsymbol{b}\|_1$.

(c) Minimize $\|\boldsymbol{Ax} - \boldsymbol{b}\|_1$ subject to $\|\boldsymbol{x}\|_\infty \le 1$.

(d) Minimize $\|\boldsymbol{x}\|_1$ subject to $\|\boldsymbol{Ax} - \boldsymbol{b}\|_\infty \le 1$.

(e) Minimize $\|\boldsymbol{Ax} - \boldsymbol{b}\|_1 + \|\boldsymbol{x}\|_\infty$.

In each of the above problems, matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ are given.

3.5 Apply each of the two methods described in Example 3.5 to solve the QP problem

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x}) = \tfrac{1}{2}(x_1^2 + x_2^2) + 2x_1 + x_2 - x_3\\ &\text{subject to: } \boldsymbol{Ax} = \boldsymbol{b} \ \text{ where } \boldsymbol{A} = [0\,1\,1], \boldsymbol{b} = 1\end{aligned}$$

3.6 Derive an explicit solution to each of the following problems:

(a)

$$\begin{aligned}&\text{minimize} \quad \boldsymbol{c}^T\boldsymbol{x}\\ &\text{subject to: } \boldsymbol{x}^T\boldsymbol{Ax} \le 1\end{aligned}$$

where $\boldsymbol{A}$ is positive definite and $\boldsymbol{c}$ is a nonzero vector.

(b)

$$\begin{aligned}&\text{minimize} \quad \boldsymbol{c}^T\boldsymbol{x}\\ &\text{subject to: } (\boldsymbol{x}-\boldsymbol{x}_c)^T\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{x}_c)\le 1\end{aligned}$$

where $\boldsymbol{A}$, $\boldsymbol{c}$, and $\boldsymbol{x}_c$ are given, $\boldsymbol{A}$ is positive definite, and $\boldsymbol{c}$ is a nonzero.

3.7 Show that the solution of the quadratic constrained quadratic programming problem

$$\begin{aligned}&\text{minimize } \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}+\boldsymbol{x}^T\boldsymbol{p}\\ &\text{subject to: } \boldsymbol{x}^T\boldsymbol{x}\le 1\end{aligned}$$

with $\boldsymbol{H}$ positive definite is given by $\boldsymbol{x}^* = -(\boldsymbol{H}+\mu\boldsymbol{I})^{-1}\boldsymbol{p}$ where $\mu = \max\{0,\bar{\mu}\}$ and $\bar{\mu}$ is the largest solution of the nonlinear equation

$$\boldsymbol{p}^T(\boldsymbol{H}+\mu\boldsymbol{I})^{-2}\boldsymbol{p}=1$$

3.8 Let $H(\omega)=\sum_{i=0}^{N} a_i\cos(i\omega)$ and $\boldsymbol{x}=\begin{bmatrix}a_0 & a_1 & \cdots & a_N\end{bmatrix}^T$. Convert the constrained optimization problem

$$\begin{aligned}&\text{minimize } f(\boldsymbol{x})=\int_0^{\pi} W(\omega)\,|H(\omega)-H_d(\omega)|^2\,d\omega\\ &\text{subject to: } |H(\omega_k)-H_d(\omega_k)|\le\delta_k \text{ for } k=1,2,\ldots,K\end{aligned}$$

into a standard QP problem and show that it is a convex QP problem. In this problem, $H_d(\omega)$ and $W(\omega)$ are given real-valued functions, $W(\omega)\ge 0$ is a weighting function, $\{\omega_k, k=1,2,\ldots,K\}$ is a set of grid points on $[0,\pi]$, and $\delta_k>0$ for $1\le k\le K$ are constants.

3.9 Show that $\boldsymbol{w}^T\boldsymbol{w}\le uv, u\ge 0$ and $v\ge 0$ if and only if

$$\left\|\begin{bmatrix}2\boldsymbol{w}\\ u-v\end{bmatrix}\right\|\le u+v$$

3.10 Solve the QP problem

$$\begin{aligned}\text{minimize } & f(x)=4x_1^2+2x_1x_2+x_2^2+2x_1+3x_2\\ \text{subject to: } & x_1\le 3\\ & -x_1+x_2\le -2\\ & x_1+x_2\le 2\end{aligned}$$

(a) By applying KKT conditions directly without using any software.

(b) Using **CVX** (include your code).

3.11 (a) Formulate the problem of solving the system of linear equations

$$\begin{aligned}x_1+2x_2&=7\\ 2x_1+x_2&=5\end{aligned}$$

$$\text{subject to: } \begin{cases}x_1-3\le 0\\ -x_2\le 0\\ -x_1+x_2-1.7\le 0\end{cases}$$

as a convex constrained problem.

(b) Solve the problem in part (a) by applying KKT conditions.

(c) Solve the problem in part (a) using **CVX**.

3.12 In the study of data interpolation by convex functions, we consider the question: Does there exist a convex function $f(\boldsymbol{x})$ with dom $f = R^n$ that satisfies $f(\boldsymbol{x}_i) = y_i$ for $i = 1, 2, \ldots, m$? It can be shown that the statement in the above question holds true if and only if there exist vectors $\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_m$ such that

$$y_j \ge y_i + \boldsymbol{g}_i^T(\boldsymbol{x}_j - \boldsymbol{x}_i) \quad \text{for } i, j = 1, \ldots, m \tag{P3.1}$$

In (P3.1), we may take vectors $\boldsymbol{g}_i$'s as gradients or subgradients of $f(\boldsymbol{x})$ at $\boldsymbol{x}_i$, depending on whether or not $f(\boldsymbol{x})$ is differentiable at $\boldsymbol{x}_i$. Furthermore, if the conditions in (P3.1) are satisfied, then a (piecewise linear) convex function $f(\boldsymbol{x})$ can be constructed as

$$f(\boldsymbol{x}) = \max_{i=1,2,\ldots,m} \{y_i + \boldsymbol{g}_i^T(\boldsymbol{x} - \boldsymbol{x}_i)\}$$

which satisfies $f(\boldsymbol{x}_i) = y_i$ for $i = 1, 2, \ldots, m$. Based on these, formulate the problem of fitting a convex function to given data $\{(\boldsymbol{x}_i, \hat{y}_i), i = 1, 2, \ldots, m\}$ as

$$\begin{aligned} &\text{minimize} \quad \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 \\ &\text{subject to: } y_j \ge y_i + \boldsymbol{g}_i^T(\boldsymbol{x}_j - \boldsymbol{x}_i) \text{ for } i, j = 1, 2, \ldots, m \end{aligned} \tag{P3.2a-b}$$

where y_i's and $\boldsymbol{g}_i$'s are treated as variables.

Express (P3.2) as a convex QP problem in standard notation i.e. as

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{p} \\ &\text{subject to: } \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b} \end{aligned}$$

3.13 Find scalars x_1, x_2, and x_3 such that the maximum eigenvalue of $\boldsymbol{F} = \boldsymbol{A}_0 + x_1\boldsymbol{A}_1 + x_2\boldsymbol{A}_2 + x_3\boldsymbol{A}_3$ with

$$\boldsymbol{A}_0 = \begin{bmatrix} 2 & -0.5 & -0.6 \\ -0.5 & 2 & 0.4 \\ -0.6 & 0.4 & 3 \end{bmatrix}, \; \boldsymbol{A}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \boldsymbol{A}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

is minimized.

3.14 Consider matrix $\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{A}_0 + x_1\boldsymbol{A}_1 + \ldots + x_n\boldsymbol{A}_n$ which depends on parameter vector $\boldsymbol{x} = [x_1\ x_2\ \ldots\ x_n]^T$ affinely. The 2-norm of $\boldsymbol{A}(\boldsymbol{x})$ is defined by $\|\boldsymbol{A}(\boldsymbol{x})\|_2 = \lambda_{\max}\left(\boldsymbol{A}(\boldsymbol{x})^T\boldsymbol{A}(\boldsymbol{x})\right)$. We are interested in finding an $\boldsymbol{x}$ that minimizes $\|\boldsymbol{A}(\boldsymbol{x})\|_2$. Formulate the problem as an SDP problem.

3.15 A classic way to build a local quadratic model for a given function in a vicinity of a given point (typically the kth iterate $\boldsymbol{x}_k$ from an algorithm) is to use the Taylor expansion of the function at the given point up to its second-order term. But what if the function in that local region is not convex? Even worse, what if explicit description of the function in question is not available? The aim of this exercise problem is to develop a different (and contemporary) approach to building a local convex quadratic model based on a set of data $\{\boldsymbol{x}^{(i)}, y^{(i)}, i = 1, 2, \ldots, m\}$ where points $\boldsymbol{x}^{(i)}$ are collected from the small region centered around a given point $\boldsymbol{x}_k$, i.e. $\boldsymbol{x}^{(i)} = \boldsymbol{x}_k + \boldsymbol{\delta}_i$ with $\boldsymbol{\delta}_i$ small in magnitude; and $y^{(i)} = f(\boldsymbol{x}^{(i)})$ for some function $f(\boldsymbol{x})$, however $f(\boldsymbol{x})$ is *not* explicitly available. Under these circumstances, a local convex quadratic model assumes the form

$$f(\boldsymbol{x}) = \tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}_k) + (\boldsymbol{x} - \boldsymbol{x}_k)^T\boldsymbol{p} + c$$

with $\boldsymbol{H} \succeq \boldsymbol{0}$. The model parameters, namely $\boldsymbol{H}$, $\boldsymbol{p}$, and c, need to be estimated for the model to be useful. To this end, one may employ a least-squares approach to determine $\boldsymbol{H}$, $\boldsymbol{p}$, and c by solving the convex problem

$$\begin{aligned}&\underset{\boldsymbol{H},\boldsymbol{p},c}{\text{minimize}} \quad \sum_{i=1}^{m}\left[\tfrac{1}{2}(\boldsymbol{x}^{(i)}-\boldsymbol{x}_k)^T\boldsymbol{H}(\boldsymbol{x}^{(i)}-\boldsymbol{x}_k)+(\boldsymbol{x}^{(i)}-\boldsymbol{x}_k)^T\boldsymbol{p}+c-y^{(i)}\right]^2\\ &\text{subject to: } \boldsymbol{H}\succeq\boldsymbol{0}\end{aligned}$$

which is obviously equivalent to

$$\begin{aligned}&\underset{\boldsymbol{H},\boldsymbol{p},c}{\text{minimize}} \quad \sum_{i=1}^{m}\left(\tfrac{1}{2}\boldsymbol{\delta}_i^T\boldsymbol{H}\boldsymbol{\delta}_i+\boldsymbol{\delta}_i^T\boldsymbol{p}+c-y^{(i)}\right)^2\\ &\text{subject to: } \boldsymbol{H}\succeq\boldsymbol{0}\end{aligned} \tag{P3.3}$$

You are asked to use MATLAB and **cvx** to carry out a small scale simulation study that is related to the above modeling problem. In the simulations, $\boldsymbol{H}$ is a 5 by 5 matrix and the size of the data set is $m = 600$.

(a) Use the code below to generate data $\{\boldsymbol{\delta}^{(i)}, y^{(i)}, i = 1, 2, \ldots, m\}$ with $m = 600$:

```
randn('state',25)
Hd = randn(5,5);
Hd = Hd'*Hd;
pd = 2*randn(5,1);
cd = randn(1) - 1;
D = 0.15*randn(5,600);
e = ones(m,1);
randn('state',37)
y = 0.5*diag(D'*Hd*D) + D'*pd + cd*e + 0.1*randn(600,1);
```

Note: In the above data set, $\boldsymbol{\delta}^{(i)}$ is the ith column of matrix D, and $y^{(i)}$ is the ith componet of vector $\boldsymbol{y}$.

(b) With the data set produced in part (a), use cvx to solve the convex problem in (P3.3) to get model parameters $\boldsymbol{H}$, $\boldsymbol{p}$, and c. You are required to provide your cvx code and the numerical values of $\boldsymbol{H}$, $\boldsymbol{p}$, and c obtained.

3.16 (*Robust LP*) The robust LP refers to a class of LP problems

$$\begin{aligned}&\text{minimize} \quad \boldsymbol{c}^T\boldsymbol{x}\\ &\text{subject to: } \boldsymbol{a}_i^T\boldsymbol{x}\le b_i \text{ for } i=1,2,\ldots,q\end{aligned} \tag{P3.4a-b}$$

where each $\boldsymbol{a}_i \in \mathcal{E}_i = \{\overline{\boldsymbol{a}}_i + \boldsymbol{P}_i\boldsymbol{u} : \|\boldsymbol{u}\|_2 \le 1\}$. In other words, $\boldsymbol{a}_i$'s contain uncertainties. We seek to find a solution $\boldsymbol{x}^*$ that solves (P3.4) in the worst case that ensures the satisfaction of the q constraints while each $\boldsymbol{a}_i$ varies within $\mathcal{E}_i$ such that $\boldsymbol{a}_i^T\boldsymbol{x}$ reaches its largest value. Formulate the problem at hand as an SOCP problem.

3.17 There is a stochastic model for robust LP problems in (P3.4) where each $\boldsymbol{a}_i$ is a random variable and the constraints in (P3.4b) must hold with probability η (typically η is set to 0.95, 0.99, or 0.999). In this way, the robust LP problem can be expressed as

$$\begin{aligned}&\text{minimize} \quad \boldsymbol{c}^T\boldsymbol{x}\\ &\text{subject to: } \text{prob}(\boldsymbol{a}_i^T\boldsymbol{x}\le b_i)\ge\eta \text{ for } i=1,2,\ldots,q\end{aligned} \tag{P3.5a-b}$$

By assuming $\boldsymbol{a}_i$'s are Gaussian random variables, formulate problem (P3.5) as an SOCP problem.

3.18 The purpose of this exercise problem is solve a digital filter design problem by formulating and solving it as an SOCP problem. A finite-impulse-response (FIR) digital filetr of length N is characterized by its transfer function given by

$$H(z) = \sum_{n=0}^{N-1} h_n z^{-n}$$

A typical filter design problem is carried out in the frequency domain where the frequency response of the filter to be designed is given by

$$H(e^{j\omega}) = \sum_{n=0}^{N-1} h_n e^{-jn\omega}$$

with the normalized frequency $\omega \in [0, \pi]$, and the design problem is to determine real-valued filter coefficients $\{h_n,\ n = 0, 1, \ldots, N - 1\}$ such that $H(e^{j\omega})$ best approximates a *desired* frequency response (e.g., an ideal lowpass frequency response) denoted by $H_d(e^{j\omega})$ in a certain sense. In the design problem being considered here, $H_d(e^{j\omega})$ is an ideal bandpass frequency response with lower passband group delay:

$$H_d(e^{j\omega}) = \begin{cases} 0 & \text{for } \omega \in [0, \omega_{a1}] \\ e^{-jK\omega} & \text{for } \omega \in [\omega_{p1}, \omega_{p2}] \\ 0 & \text{for } \omega \in [\omega_{a2}, \pi] \end{cases}$$

where K is a passband group delay, $\omega_{a1} = 0.32\pi$, $\omega_{p1} = 0.35\pi$, $\omega_{p2} = 0.65\pi$, and $\omega_{a2} = 0.68\pi$, and the design goal is to find $\{h_n,\ n = 0, 1, \ldots, N - 1\}$ such that maximum approximation error over the union of the three frequency bands $\Omega = [0, 0.32\pi] \cup [0.35\pi, 0.65\pi] \cup [0.68\pi, \pi]$ is minimized. Realistically, the optimization will be carried out over a finite number M of frequency grids, Ω_d, that are evenly placed over Ω. Under these circumstances, the optimization problem becomes

$$\underset{\{h_n, n=0,1,\ldots,N-1\}}{\text{minimize}} \ \max_{\omega \in \Omega_d} \ \left| H(e^{j\omega}) - H_d(e^{j\omega}) \right|$$

(a) Formulate the problem as a standard SOCP problem where all terms involved need to be explicitly described.

(b) Use **`CVX`** to solve the SOCP problem from part (a) with $N = 135$, $K = 64$, and $M = 500$.

(c) Include your **`CVX`** code and report your design result by displaying the amplitude response (in dB) of the filter obtained. This can be readily done using MATLAB command **`freqz`**.

Chapter 4 Algorithms for General Convex Problems

4.1 Introduction

In Chapter 3, we have addressed several special classes of convex problems that have found applications in engineering disciplines. There are however many problems in practice that do not fall into these classes. This chapter is devoted to studies of methods for general convex problems. These include Newton algorithms, proximal-point algorithms for composite convex functions, and alternating direction algorithms.

Several concepts and properties of convex functions that are needed in the development of these algorithms are introduced in Sec. 4.2. These include subgradients, convex functions with Lipschitz continuous gradients, strongly convex functions, conjugate functions, and proximal operators. Newton algorithms for unconstrained and constrained convex problems are addressed in Sec. 4.3. In Sec. 4.4, several algorithms for minimizing composite convex functions are studied. The l_1-l_2 minimization problem which finds applications in digital signal processing and machine learning is covered as a special case. Alternating direction methods have become increasingly important because of their ability to deal with large scale convex problems. In Sec. 4.5, we present two representative classes of alternating direction methods known as alternating direction methods of multipliers and alternating minimization algorithms.

4.2 Concepts and Properties of Convex Functions

The notion of convex functions and their elementary properties are addressed in Chapter 2. Here we introduce several additional concepts and properties of convex functions that are of use in the development of effective algorithms for convex problems.

4.2.1 Subgradients

Practical optimization problems involving non-differentiable objective functions and/or constraints are pervasive. The non-smoothness of the functions in an optimization problem may be encountered in several ways. For example, there are optimization problems with objective functions or constraints that are inherently non-differentiable. A simple example of the case is given by

$$\begin{array}{ll}\text{minimize} & \|\boldsymbol{x}\|_1 \\ \text{subject to:} & \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2 \le \varepsilon\end{array}$$

where the l_1-norm of variable $\boldsymbol{x}$ is minimized subject to an l_2-norm constraint. Obviously, the objective function $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$ is continuous and convex, but *not differentiable*. There are also scenarios where the functions involved are differentiable, and it is the operation of these functions that yields non-smoothness. An example of the case is the objective function of the form

$$f(\boldsymbol{x}) = \max_{1\le j\le p}\{\varphi_j(\boldsymbol{x})\}$$

where $\varphi_j(\boldsymbol{x})$ are smooth convex functions, and the objective function is also convex but not necessarily differentiable, see Fig. 4.1 that depicts the objective function with three linear $\varphi_j(\boldsymbol{x})$.

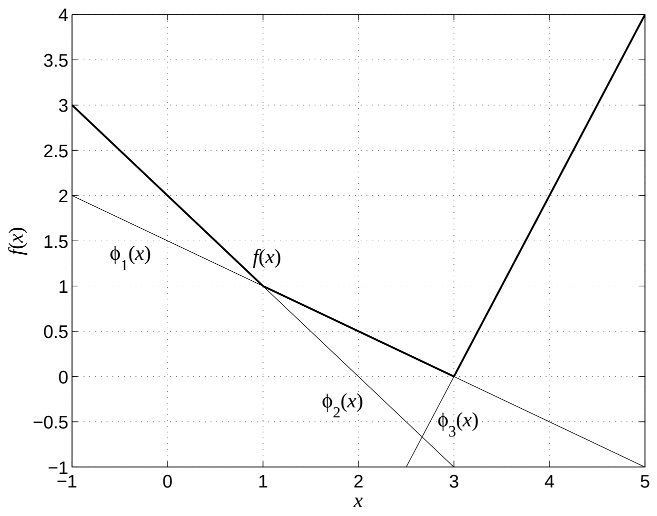


Figure 4.1. Pointwise maximum of three affine functions yields a piece-wise affine function which is convex but not differentiable.

Gradient plays an instrumental role in continuous optimization for functions that are differentiable. The concept of *subgradient* is a natural generalization of the concept of gradient that allows us to deal with optimization problems involving convex but non-differentiable functions [1]. Recall that the convexity of a differentiable function $f(\boldsymbol{x})$ can be characterized by

$$f(\tilde{\boldsymbol{x}}) \geq f(\boldsymbol{x}) + \nabla^T f(\boldsymbol{x})(\tilde{\boldsymbol{x}} - \boldsymbol{x}) \qquad \text{for } \boldsymbol{x}, \tilde{\boldsymbol{x}} \in \text{dom}(f) \tag{4.1}$$

where $\text{dom}(f)$ is the domain of function $f(\boldsymbol{x})$ that defines the set of points $\boldsymbol{x}$ where $f(\boldsymbol{x})$ assumes finite values. In geometric terms, Eq. (4.1) states that at any point $\boldsymbol{x}$ in the domain of a convex function $f(\boldsymbol{x})$, the tangent to the surface defined by $y = f(\boldsymbol{x})$ always lies below the surface.

Definition 4.1 If $f(\boldsymbol{x})$ is convex but not necessarily differentiable, then vector $\boldsymbol{g} \in R^n$ is said to be a *subgradient* of $f(\boldsymbol{x})$ at $\boldsymbol{x}$ if

$$f(\tilde{\boldsymbol{x}}) \geq f(\boldsymbol{x}) + \boldsymbol{g}^T (\tilde{\boldsymbol{x}} - \boldsymbol{x}) \quad \text{for } \tilde{\boldsymbol{x}} \in \text{dom}(f) \tag{4.2}$$

The subgradient of a convex function at point $\boldsymbol{x}$ where $f(\boldsymbol{x})$ is non-differentiable is *not* unique. The set of all subgradients at point $\boldsymbol{x}$ is called *subdifferential* of $f(\boldsymbol{x})$ and is denoted by $\partial f(\boldsymbol{x})$. ■

The right-hand side of Eq. (4.2) may be viewed as a linear lower bound of $f(\boldsymbol{x})$, and the subgradients at a point $\boldsymbol{x}$ where the convex function $f(\boldsymbol{x})$ is not differentiable correspond to different tangent lines at $\boldsymbol{x}$. This is illustrated in Fig. 4.2, where the two subgradients of $f(x)$ at x^* are given by $g_1 = \tan\theta_1$ and $g_2 = \tan\theta_2$. From the figure, it is obvious that any tangent line at x^* with a slop between g_2 and g_1 satisfies Eq. (4.2), therefore, any value $g \in [g_2, g_1]$ is a subgradient of $f(x)$ at x^*.

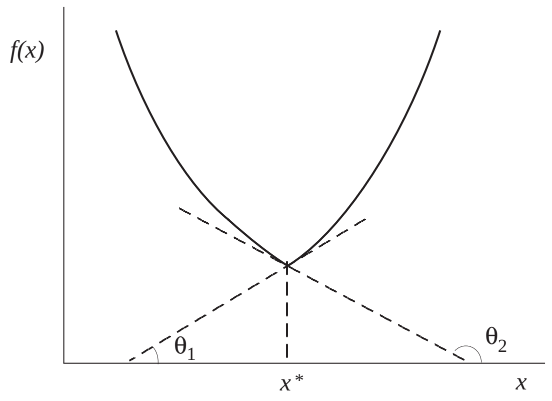


Figure 4.2. Two subgradients of $f(x)$ at x^* where $f(x)$ is not differentiable.

From Eq. (4.2), it follows that $f(\tilde{\boldsymbol{x}}) \geq f(\boldsymbol{x})$ as long as $\boldsymbol{g}^T(\tilde{\boldsymbol{x}} - \boldsymbol{x}) \geq 0$. Note that for a given point $\boldsymbol{x}$, $\boldsymbol{g}^T(\tilde{\boldsymbol{x}} - \boldsymbol{x}) = 0$ defines a hyperplane which passes through point $\boldsymbol{x}$ with $\boldsymbol{g}$ as its normal. This hyperplane divides space R^n into two parts with the hyperplane as boundary. In the part of the space where $\tilde{\boldsymbol{x}}$ satisfies $\boldsymbol{g}^T(\tilde{\boldsymbol{x}} - \boldsymbol{x}) > 0$, no minimizers exist because Eq. (4.2) in this case implies that $f(\tilde{\boldsymbol{x}}) > f(\boldsymbol{x})$. Consequently, a minimizer of $f(\boldsymbol{x})$ can only be found in the part of the space characterized by $\{\boldsymbol{x} : \boldsymbol{g}^T(\tilde{\boldsymbol{x}} - \boldsymbol{x}) \leq 0\}$. In this analysis, we see subgradient facilitates the construction of a cutting plane in the parameter space that reduces the search region significantly.

There are several important special cases in which the computation of a subgradient of a convex $f(\boldsymbol{x})$ can be readily carried out:

(a) If $f(\boldsymbol{x})$ is convex and differentiable at $\boldsymbol{x}$, then the subdifferential $\partial f(\boldsymbol{x})$ contains only one subgradient which is the same as the gradient $\nabla f(\boldsymbol{x})$;

(b) If $\alpha > 0$, a subgradient of $\alpha f(\boldsymbol{x})$ is given by $\alpha\,\boldsymbol{g}$ where $\boldsymbol{g}$ is a subgradient of $f(\boldsymbol{x})$;

(c) If $f(\boldsymbol{x}) = f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}) + \cdots + f_r(\boldsymbol{x})$ where $f_i(\boldsymbol{x})$ for $1 \leq i \leq r$ are convex, then $\boldsymbol{g} = \boldsymbol{g}_1 + \boldsymbol{g}_2 + \cdots + \boldsymbol{g}_r$ is a subgradient of $f(\boldsymbol{x})$ where $\boldsymbol{g}_i$ is a subgradient of $f_i(\boldsymbol{x})$;

(d) Define function $f(\boldsymbol{x}) = \max\left[f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_r(\boldsymbol{x})\right]$ where $f_i(\boldsymbol{x})$ for i = 1, 2, …, r are convex. At a given point $\boldsymbol{x}$, there exists at least one index, say i^*, such that $f(\boldsymbol{x}) = f_{i^*}(\boldsymbol{x})$. Then a subgradient of $f_{i^*}(\boldsymbol{x})$ is a subgradient of $f(\boldsymbol{x})$.

(e) If $h(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$ where $f(\boldsymbol{x})$ is convex, then

$$\partial h(\boldsymbol{x}) = \boldsymbol{A}^T \partial f(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b})$$

Example 4.1 Verify the formula of subdifferential of function $f(x) = |x|$

$$\partial |x| = \begin{cases} 1 & \text{for } x > 0 \\ \text{any } g \in [-1, 1] & \text{for } x = 0 \\ -1 & \text{for } x < 0 \end{cases} \tag{4.3}$$

Solution Function $f(x) = |x|$ is convex because for $0 \le \alpha \le 1$ we have

$$f(\alpha x_1 + (1-\alpha)x_2) = |\alpha x_1 + (1-\alpha)x_2| \le \alpha |x_1| + (1-\alpha)|x_2| = \alpha f(x_1) + (1-\alpha) f(x_2)$$

Consider a point x > 0, we have $f(x)$ = x which is obviously differentiable. Hence the differential of $f(x)$ is equal to the derivative of $f(x)$, i.e., $f^{'}(x) = x' = 1$ which verifies the first line of the formula. Now consider a point x < 0, we have $f(x)$ = $-x$ which is differentiable. Hence the differential of $f(x)$ is equal to $f^{'}(x) = (-x)' = -1$ which verifies the third line of the formula. At x = 0, Eq. (4.2) is reduced to $|\tilde{x}| \ge g\,\tilde{x}$ which holds for any g between –1 and 1. This verifies the second line of the formula in Eq. (4.3). ■

The next two theorems concern optimization problems where the functions involved are convex but not necessarily differentiable. These theorems may be regarded as extensions of the well-known first-order optimality condition and KKT conditions to their non-differentiable counterparts that are studied earlier in Chapter 2 and Chapter 10, respectively.

Theorem 4.1 ***First-order optimality condition for non-differentiable unconstrained convex problems*** Point $\boldsymbol{x}^*$ is a global solution of the minimization problem

$$\underset{\boldsymbol{x} \in \text{dom}(f)}{\text{minimize}} \; f(\boldsymbol{x})$$

where $f(\boldsymbol{x})$ is convex, if and only if $\boldsymbol{0} \in \partial f(\boldsymbol{x}^*)$.

Proof Suppose $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. By letting $\boldsymbol{x} = \boldsymbol{x}^*$ and $\boldsymbol{g} = \mathbf{0}$ in Eq. (4.2), we obtain $f(\tilde{\boldsymbol{x}}) \ge f(\boldsymbol{x}^*)$ for all $\tilde{\boldsymbol{x}}$ in $\text{dom}(f)$, hence $\boldsymbol{x}^*$ is a global minimizer of $f(\boldsymbol{x})$. Conversely, if $\boldsymbol{x}^*$ is a global minimizer, then $f(\tilde{\boldsymbol{x}}) \ge f(\boldsymbol{x}^*)$ for all $\tilde{\boldsymbol{x}} \in \text{dom}(f)$, hence $f(\tilde{\boldsymbol{x}}) \ge f(\boldsymbol{x}^*) + \mathbf{0}^T(\tilde{\boldsymbol{x}} - \boldsymbol{x}^*)$ for all $\tilde{\boldsymbol{x}} \in \text{dom}(f)$, which implies that $\mathbf{0} \in \partial f(\boldsymbol{x}^*)$. ■

Theorem 4.2 ***KKT conditions for non-differentiable constrained convex problems*** Consider constrained convex problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to: } \boldsymbol{a}_i^T \boldsymbol{x} = b_i \quad \text{for } 1 \le i \le p \\ &\qquad\qquad c_j(\boldsymbol{x}) \le 0 \quad \text{for } 1 \le j \le q \end{aligned} \tag{4.4a-c}$$

where $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$ are convex but not necessarily differentiable. A regular point $\boldsymbol{x}^*$ is a solution of the problem in Eq. (4.4) if and only if

(a) $\boldsymbol{a}_i^T \boldsymbol{x}^* = b_i$ for $1 \le i \le p$.

(b) $c_j(\boldsymbol{x}^*) \le 0$ for $1 \le j \le q$.

(c) there exist λ_i^* and μ_j^* such that $\mathbf{0} \in \partial f(\boldsymbol{x}^*) + \sum_{i=1}^{p} \lambda_i^* \boldsymbol{a}_i + \sum_{j=1}^{q} \mu_j^* \partial c_j(\boldsymbol{x}^*)$

(d) $\mu_j^* c_j(\boldsymbol{x}^*) = 0$ for $j = 1, 2, \ldots, q$.

(e) $\mu_j^* \ge 0$ for $j = 1, 2, \ldots, q$.

Proof Below we prove the sufficiency of these conditions for point $\boldsymbol{x}^*$ to be a global solution of the problem in Eq. (4.4), and leave the necessity part to the reader.
Suppose point $\boldsymbol{x}^*$ satisfies conditions (a) – (e). Conditions (a) and (b) imply that $\boldsymbol{x}^*$ is a feasible point of Eq. (4.4). Let $\boldsymbol{x}$ be an arbitrary feasible point for the problem in Eq. (4.4), below we show that $f(\boldsymbol{x}^*) \le f(\boldsymbol{x})$.

Since both $\boldsymbol{x}$ and $\boldsymbol{x}^*$ are feasible, we can write

$$\sum_{i=1}^{p} \lambda_i^* \boldsymbol{a}_i^T (\boldsymbol{x} - \boldsymbol{x}^*) = \sum_{i=1}^{p} \lambda_i^* (\boldsymbol{a}_i^T \boldsymbol{x} - b_i) - \sum_{i=1}^{p} \lambda_i^* (\boldsymbol{a}_i^T \boldsymbol{x}^* - b_i) = 0 \tag{4.5}$$

Because $c_j(\boldsymbol{x}) \le 0$ and $\mu_j^* \ge 0$ for j = 1, 2, …, q, we have

$$\sum_{j=1}^{q} \mu_j^* c_j(\boldsymbol{x}) \le 0$$

which in conjunction with condition (d) implies that

$$\sum_{j=1}^{q} \mu_j^* \left(c_j(\boldsymbol{x}) - c_j(\boldsymbol{x}^*) \right) \le 0 \tag{4.6}$$

The convexity of functions $c_j(\boldsymbol{x})$ gives

$$c_j(\boldsymbol{x}) - c_j(\boldsymbol{x}^*) \ge \partial c_j(\boldsymbol{x}^*)^T (\boldsymbol{x} - \boldsymbol{x}^*)$$

which leads Eq. (4.6) to

$$\sum_{j=1}^{q} \mu_j^* \partial c_j(\boldsymbol{x}^*)^T (\boldsymbol{x} - \boldsymbol{x}^*) \le 0 \tag{4.7}$$

From Eqs. (4.5), (4.7), and the convexity of $f(\boldsymbol{x})$, we deduce

$$\begin{aligned}
&f(\boldsymbol{x}) - f(\boldsymbol{x}^*) \\
&\ge \partial f(\boldsymbol{x}^*)^T (\boldsymbol{x} - \boldsymbol{x}^*) + \sum_{i=1}^{p} \lambda_i^* \boldsymbol{a}_i^T (\boldsymbol{x} - \boldsymbol{x}^*) + \sum_{j=1}^{q} \mu_j^* \partial c_j(\boldsymbol{x}^*)^T (\boldsymbol{x} - \boldsymbol{x}^*) \\
&= \left[\partial f(\boldsymbol{x}^*) + \sum_{i=1}^{p} \lambda_i^* \boldsymbol{a}_i + \sum_{j=1}^{q} \mu_j^* \partial c_j(\boldsymbol{x}^*) \right]^T (\boldsymbol{x} - \boldsymbol{x}^*)
\end{aligned}$$

By condition (c), the expression in the last square bracket can be set to zero which leads to $f(\boldsymbol{x}^*) \le f(\boldsymbol{x})$. ■

4.2.2 Convex functions with Lipschitz-continuous gradients

A continuously differentiable function $f(\boldsymbol{x})$ is said to have Lipschitz continuous gradient [2] if

$$\left\| \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}) \right\|_2 \le L \left\| \boldsymbol{x} - \boldsymbol{y} \right\|_2 \tag{4.8}$$

for any $\boldsymbol{x}$ and $\boldsymbol{y} \in \text{dom}(f)$, where $L > 0$ is called Lipschitz constant.

Example 4.2 Show that the gradient of $f(\boldsymbol{x}) = \frac{1}{2} \| \boldsymbol{Ax} - \boldsymbol{b} \|_2^2$ is Lipschitz continuous.

Solution We write function $f(\boldsymbol{x})$ as

$$f(\boldsymbol{x}) = \tfrac{1}{2} \boldsymbol{x}^T \boldsymbol{A}^T \boldsymbol{Ax} - \boldsymbol{x}^T \boldsymbol{A}^T \boldsymbol{b} + \tfrac{1}{2} \boldsymbol{b}^T \boldsymbol{b}$$

Hence the gradient of $f(\boldsymbol{x})$ is given by

$$\nabla f(\boldsymbol{x}) = \boldsymbol{A}^T \boldsymbol{Ax} - \boldsymbol{A}^T \boldsymbol{b}$$

and

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 = \left\|\boldsymbol{A}^T\boldsymbol{A}(\boldsymbol{x}-\boldsymbol{y})\right\|_2 \le \lambda_{\max}(\boldsymbol{A}^T\boldsymbol{A})\|\boldsymbol{x}-\boldsymbol{y}\|_2$$

where $\lambda_{\max}(\boldsymbol{A}^T\boldsymbol{A})$ denotes the largest eigenvalue of $\boldsymbol{A}^T\boldsymbol{A}$. Therefore, $\nabla f(\boldsymbol{x})$ is Lipschitz continuous with $L = \lambda_{\max}(\boldsymbol{A}^T\boldsymbol{A})$. ■

Now recall the property of a convex function $f(\boldsymbol{x})$ expressed in Eq. (4.2). Essentially it says that a convex function is bounded from below by its supporting plans. The theorem below shows that a convex function with Lipschitz continuous gradient is, in addition, bounded from above by a family of simple convex quadratic functions.

Theorem 4.3 ***A property of convex functions with Lipschitz continuous gradients*** If $f(\boldsymbol{x})$ is convex with Lipschitz continuous gradient, then $f(\boldsymbol{x})$ is bounded from above by a family of convex quadratic functions with Hessian $L\cdot\boldsymbol{I}$, namely,

$$f(\boldsymbol{x}) \le f(\boldsymbol{y}) + \nabla f(\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y}) + \frac{L}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \tag{4.9}$$

Proof The proof given below is from [2]. By expressing $f(\boldsymbol{x})$ as

$$\begin{aligned} f(\boldsymbol{x}) &= f(\boldsymbol{y}) + \int_0^1 \nabla f(\boldsymbol{y}+\tau(\boldsymbol{x}-\boldsymbol{y}))^T\,(\boldsymbol{x}-\boldsymbol{y})\,d\tau \\ &= f(\boldsymbol{y}) + \nabla f(\boldsymbol{y})^T\,(\boldsymbol{x}-\boldsymbol{y}) + \int_0^1 [\nabla f(\boldsymbol{y}+\tau(\boldsymbol{x}-\boldsymbol{y})) - \nabla f(\boldsymbol{y})]^T(\boldsymbol{x}-\boldsymbol{y})\,d\tau \end{aligned}$$

we have

$$\begin{aligned} &\left|f(\boldsymbol{x}) - f(\boldsymbol{y}) - \nabla^T f(\boldsymbol{y})\,(\boldsymbol{x}-\boldsymbol{y})\right| \\ &= \left|\int_0^1 [\nabla f(\boldsymbol{y}+\tau(\boldsymbol{x}-\boldsymbol{y})) - \nabla f(\boldsymbol{y})]^T(\boldsymbol{x}-\boldsymbol{y})\,d\tau\right| \\ &\le \int_0^1 \left|[\nabla f(\boldsymbol{y}+\tau(\boldsymbol{x}-\boldsymbol{y})) - \nabla f(\boldsymbol{y})]^T(\boldsymbol{x}-\boldsymbol{y})\right| d\tau \\ &\le \int_0^1 \|\nabla f(\boldsymbol{y}+\tau(\boldsymbol{x}-\boldsymbol{y})) - \nabla f(\boldsymbol{y})\|_2 \cdot \|\boldsymbol{x}-\boldsymbol{y}\|_2\,d\tau \\ &\le \int_0^1 \tau L\|\boldsymbol{x}-\boldsymbol{y}\|_2^2\,d\tau = \frac{L}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \end{aligned}$$

which implies Eq. (4.9). ■

Note that the quadratic upper bound in Eq. (4.9) has a simple constant Hessian $L\cdot\boldsymbol{I}$, therefore it is straightforward to compute the minimizer of the upper bound as a function of $\boldsymbol{x}$. In effect, the upper bound in Eq. (4.9) can be written as

$$h_U(\boldsymbol{x},\boldsymbol{y}) = f(\boldsymbol{y}) + \nabla f(\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y}) + \frac{L}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 = \frac{L}{2}\left\|\boldsymbol{x} - \left(\boldsymbol{y} - \tfrac{1}{L}\nabla f(\boldsymbol{y})\right)\right\|_2^2 + \text{const}$$

Hence its global minimizer is given by

$$\hat{\boldsymbol{x}} = \boldsymbol{y} - \tfrac{1}{L}\nabla f(\boldsymbol{y}) \tag{4.10}$$

As a simple application of above analysis, suppose we are in the *k*th iteration of minimizing a convex function $f(\boldsymbol{x})$ with Lipschitz continuous gradient and the *k*th iterate $\boldsymbol{x}_k$ is available. Other than minimizing $f(\boldsymbol{x})$ itself, $\boldsymbol{x}_k$ can be updated by minimizing the upper bound of $f(\boldsymbol{x})$ in Eq. (4.9) with $\boldsymbol{y}$ taken to be $\boldsymbol{x}_k$. In doing so, Eq. (4.10) can be used to generate the next iterate $\boldsymbol{x}_{k+1}$ as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \tfrac{1}{L}\nabla f(\boldsymbol{x}_k) \tag{4.11}$$

It is interesting to note that Eq. (4.11) is a steepest descent formula with a *constant* line search step $\alpha_k = 1/L$.

The properties in Eqs. (4.2) and (4.9) can be combined to write

$$f(\boldsymbol{y}) + \nabla f(\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y}) \le f(\boldsymbol{x}) \le f(\boldsymbol{y}) + \nabla f(\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y}) + \frac{L}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2 \tag{4.12}$$

Example 4.3

(a) Verify the property in Eq. (4.12) by graphics for the scalar-variable function $f(x) = \tfrac{1}{2}(x-1)^2 - \tfrac{1}{2}\log(x^2+1) + 3$ where y is set to 3.

(b) With an initial guess $x_0 = 3$, apply the formula in Eq. (4.11) to generate three iterates and evaluate how close they are to the minimizer of $f(x)$.

Solution (a) The first- and second-order derivatives of $f(x)$ are given by

$$f'(x) = x - 1 - \frac{x}{x^2+1} \quad \text{and} \quad f''(x) = \frac{x^2(x^2+3)}{(x^2+1)^2}$$

respectively. Since $f''(x) \ge 0$, $f(x)$ is convex. Moreover, it can be verified that

$$|f'(x) - f'(y)| \le 1.25\,|x-y|$$

hence $f'(x)$ is Lipschitz continuous with L = 1.25. At y = 3, the lower and upper bounds in Eq. (4.12) are given by

$$h_L(x,3) = 3.8487 + 1.7(x-3)$$

and

$$h_U(x,3) = 3.8487 + 1.7(x-3) + 0.625(x-3)^2$$

The above bounds as well as function $f(x)$ itself over the region [0, 6]are depicted in Fig. 4.3.

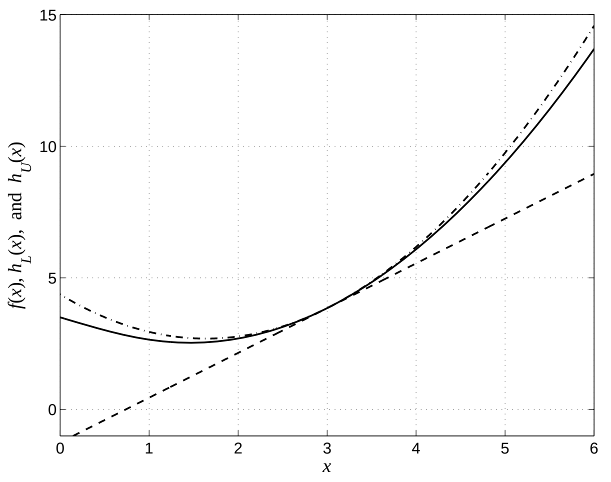


Figure 4.3. Function $f(x)$ (solid curve), its linear lower bound (dashed line) and quadratic upper bound (dot-dashed curve) over interval [0, 6].

(b) By setting $f^{'}(x) = 0$, we obtain a third-order equation $x^3 - x^2 - 1 = 0$ which has only one real root $x^* = 1.4656$. Since $f(x)$ is strictly convex, x^* is the unique global minimizer of $f(x)$. By applying Eq. (4.11) with

$$\nabla f(x_k) = x_k - 1 - \frac{x_k}{x_k^2 + 1}$$

L = 1.25 and x_0 = 3, the first three iterates were found to be x_1 = 1.64, x_2 = 1.4838, and x_3 = 1.4675 and apparently x_3 is a good approximation of minimizer x^*. ■

4.2.3 Strongly convex functions

Definition 4.2 A continuously differentiable function $f(\boldsymbol{x})$ is said to be *strongly convex* if there exists $m > 0$ such that

$$f(\boldsymbol{x}_1) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T (\boldsymbol{x}_1 - \boldsymbol{x}) + \frac{m}{2} \| \boldsymbol{x}_1 - \boldsymbol{x} \|_2^2 \tag{4.13}$$

holds for any $\boldsymbol{x}_1$ and $\boldsymbol{x}$.

As seen in the right-hand of Eq. (4.13), a strongly convex function has a quadratic lower bound which is obviously tighter than the linear lower bound given in Eq. (4.2) because the second-order term there is always nonnegative. One would therefore expect that the class of strongly convex functions possesses additional desirable properties relative to conventional convex functions. Below we introduce some of these properties [3].

Properties of strongly convex functions

(a) A twice continuously differentiable function $f(\boldsymbol{x})$ is strongly convex if and only if

$$\nabla^2 f(\boldsymbol{x}) \succeq m\boldsymbol{I} \tag{4.14}$$

for some $m > 0$.

Proof Suppose Eq. (4.14) holds. The Taylor expansion of $f(\boldsymbol{x})$ at point $\boldsymbol{x}_1$ is given by

$$f(\boldsymbol{x}_1) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{x}_1 - \boldsymbol{x}) + \tfrac{1}{2}(\boldsymbol{x}_1 - \boldsymbol{x})^T \nabla^2 f(\boldsymbol{y})(\boldsymbol{x}_1 - \boldsymbol{x}) \tag{4.15}$$

for some $\boldsymbol{y}$ on the line between $\boldsymbol{x}$ and $\boldsymbol{x}_1$, which in conjunction with Eq. (4.14) leads to Eq. (4.13). Therefore, $f(\boldsymbol{x})$ is strongly convex. Conversely, suppose $f(\boldsymbol{x})$ is strongly convex, then Eqs. (4.13) and (4.15) imply that

$$(\boldsymbol{x}_1 - \boldsymbol{x})^T \nabla^2 f(\boldsymbol{y})(\boldsymbol{x}_1 - \boldsymbol{x}) \ge m \,\| \boldsymbol{x}_1 - \boldsymbol{x} \|_2^2$$

By choosing $\boldsymbol{x}_1 = \boldsymbol{x} + t\boldsymbol{\delta}$ with t a scalar and $\boldsymbol{\delta}$ an arbitrary vector in R^n, the above expression becomes $\boldsymbol{\delta}^T[\nabla^2 f(\boldsymbol{y}) - m\boldsymbol{I}]\boldsymbol{\delta} \ge 0$ which leads to

$$\boldsymbol{\delta}^T[\nabla^2 f(\boldsymbol{x}) - m\boldsymbol{I}]\boldsymbol{\delta} \ge 0$$

as $t \to 0$. Hence $\nabla^2 f(\boldsymbol{x}) \succeq m\boldsymbol{I}$ and the proof is complete. ■

(b) If $f(\boldsymbol{x})$ is strongly convex and f^* is the minimum value of $f(\boldsymbol{x})$, then

$$f(\boldsymbol{x}) - f^* \le \frac{1}{2m} \| \nabla f(\boldsymbol{x}) \|_2^2 \tag{4.16}$$

Proof Note that the convex quadratic function of $\boldsymbol{x}_1$ on the right-hand of Eq. (4.13) achieves its minimum at $\boldsymbol{x}_1^* = \boldsymbol{x} - \frac{1}{m}\nabla f(\boldsymbol{x})$. This implies that

$$\begin{aligned} f(\boldsymbol{x}_1) &\ge f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{x}_1 - \boldsymbol{x}) + \frac{m}{2} \| \boldsymbol{x}_1 - \boldsymbol{x} \|_2^2 \\ &\ge f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{x}_1^* - \boldsymbol{x}) + \tfrac{m}{2} \| \boldsymbol{x}_1^* - \boldsymbol{x} \|_2^2 \\ &= f(\boldsymbol{x}) - \tfrac{1}{2m} \| \nabla f(\boldsymbol{x}) \|_2^2 \end{aligned} \tag{4.17}$$

Eq. (4.17) also holds when $f(\boldsymbol{x}_1)$ is replaced by its infimun, i.e., $f^* \ge f(\boldsymbol{x}) - \frac{1}{2m} \| \nabla f(\boldsymbol{x}) \|_2^2$, hence Eq. (4.16). ■

The upper bound in Eq. (4.16) quantifies how the closeness of $f(\boldsymbol{x})$ to its minimum can be measured in terms of its gradient. In effect, from Eq. (4.16) we can deduce

$$f(\boldsymbol{x}) - f^* \le \varepsilon \quad \text{if} \quad \| \nabla f(\boldsymbol{x}) \|_2 \le \sqrt{2m\varepsilon}$$

The next property provides an upper bound on the closeness of a point $\boldsymbol{x}$ to a minimizer $\boldsymbol{x}^*$ in terms of its gradient.

(c) If $f(\boldsymbol{x})$ is strongly convex and $\boldsymbol{x}^*$ is a minimizer with $f(\boldsymbol{x}^*) = f^*$, then

$$\| \boldsymbol{x}-\boldsymbol{x}^* \|_2 \le \frac{2}{m} \| \nabla f(\boldsymbol{x}) \|_2 \tag{4.18}$$

Proof By Eq. (4.13) with $\boldsymbol{x}_1 = \boldsymbol{x}^*$, we have

$$\nabla f(\boldsymbol{x})^T (\boldsymbol{x}^* - \boldsymbol{x}) + \frac{m}{2} \| \boldsymbol{x}^* - \boldsymbol{x} \|_2^2 \le f(\boldsymbol{x}^*) - f(\boldsymbol{x}) \le 0$$

Hence

$$\frac{m}{2} \| \boldsymbol{x}^* - \boldsymbol{x} \|_2^2 \le -\nabla f(\boldsymbol{x})^T (\boldsymbol{x}^* - \boldsymbol{x}) \le \| \nabla f(\boldsymbol{x}) \|_2 \cdot \| \boldsymbol{x}^* - \boldsymbol{x} \|_2$$

which yields Eq. (4.18). ■

(d) If $f(\boldsymbol{x})$ is strongly convex, then

$$\left(\nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x})\right)^T (\boldsymbol{x}_1 - \boldsymbol{x}) \ge m \| \boldsymbol{x}_1 - \boldsymbol{x} \|_2^2 \tag{4.19}$$

Proof By Eq. (4.13), we can write

$$f(\boldsymbol{x}) \ge f(\boldsymbol{x}_1) + \nabla f(\boldsymbol{x}_1)^T (\boldsymbol{x} - \boldsymbol{x}_1) + \frac{m}{2} \| \boldsymbol{x} - \boldsymbol{x}_1 \|_2^2$$

Adding Eq. (4.13) to the above expression yields

$$\nabla f(\boldsymbol{x})^T (\boldsymbol{x}_1 - \boldsymbol{x}) + \nabla f(\boldsymbol{x}_1)^T (\boldsymbol{x} - \boldsymbol{x}_1) + m \| \boldsymbol{x} - \boldsymbol{x}_1 \|_2^2 \le 0$$

which implies Eq. (4.19). ■

If $f(\boldsymbol{x})$ is strongly convex and possesses Lipschitz continuous gradient (see Sec. 4.2.2), then Eqs. (4.8) and (4.19) imply that

$$m \| \boldsymbol{x} - \boldsymbol{y} \|_2^2 \le (\boldsymbol{x} - \boldsymbol{y})^T \left(\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\right) \le L \| \boldsymbol{x} - \boldsymbol{y} \|_2^2$$

The value $L/m \ge 1$ is called the *condition number* of function $f(\boldsymbol{x})$.

4.2.4 Conjugate functions

An important notion in convex analysis is that of conjugate functions [1]. We start by introducing several concepts relevant to conjugate functions.

Definition 4.3 The *epigraph* of a real-valued function $f(\boldsymbol{x})$ is defined as the set

$$\text{epi}(f) = \{(\boldsymbol{x}, u) : \boldsymbol{x} \in \text{dom}(f), u \ge f(\boldsymbol{x})\}$$

It can be shown [2] that function $f(\boldsymbol{x})$ is convex if and only if set $\text{epi}(f)$ is convex. Fig. 4.4 illustrates the connection between the convexity of $\text{epi}(f)$ and the convexity of $f(\boldsymbol{x})$.

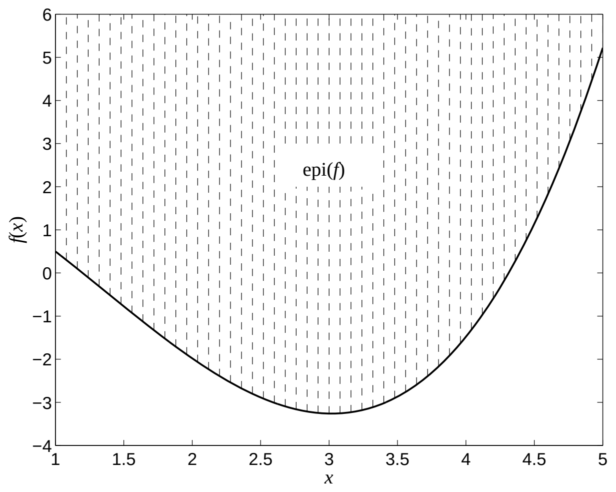


Figure 4.4. Epigraph of a convex function.

Definition 4.4 [4] Function $f(\boldsymbol{x})$ is said to be *lower-semicontinuous* at $\boldsymbol{x}_0$ if for every $\varepsilon > 0$ there exists a neighborhood $\mathcal{U}$ of $\boldsymbol{x}_0$ such that $f(\boldsymbol{x}) \geq f(\boldsymbol{x}_0) - \varepsilon$ for all $\boldsymbol{x} \in \mathcal{U}$.

Fig. 4.5 illustrates the concept of lower-semi continuity where at $\boldsymbol{x}_0$ the function in Fig. 4.5(a) is lower-semicontinuous, while the function in Fig. 4.5(b) is not.

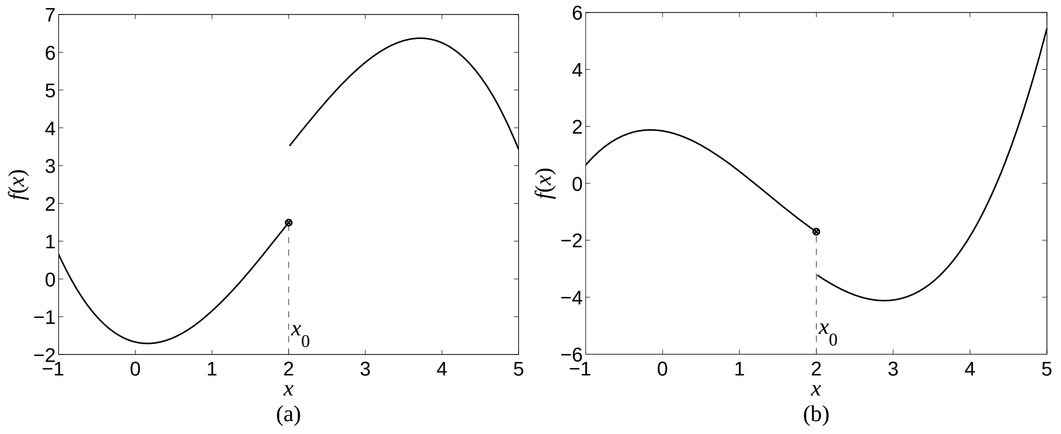


Figure 4.5. (a) A function that is lower-semicontinuous at x_0, and (b) a function that is not lower-semicontinuous at x_0.

It can be shown that a function is lower-semicontinuous if its epigraph is a closed set [1]. This is illustrated in Fig. 4.6 where the epi(f) of a lower-semicontinuous function shown in Fig. 4.6(a) is a closed set while the epi(f) of a function that is not lower-semicontinuous shown in Fig. 4.6(b) is not a closed set.

Definition 4.5 Function $f(\boldsymbol{x})$ is said to be *proper* if dom(f) is nonempty.

We recall that by definition the domain of $f(\boldsymbol{x})$ is defined by $\text{dom}(f) = \{\boldsymbol{x} : f(\boldsymbol{x}) < +\infty\}$. It is customary that a proper function $f(\boldsymbol{x})$ is extended to the entire space R^n such that at any point $\boldsymbol{x}$ outside $\text{dom}(f)$ $f(\boldsymbol{x}) = +\infty$ is assumed.

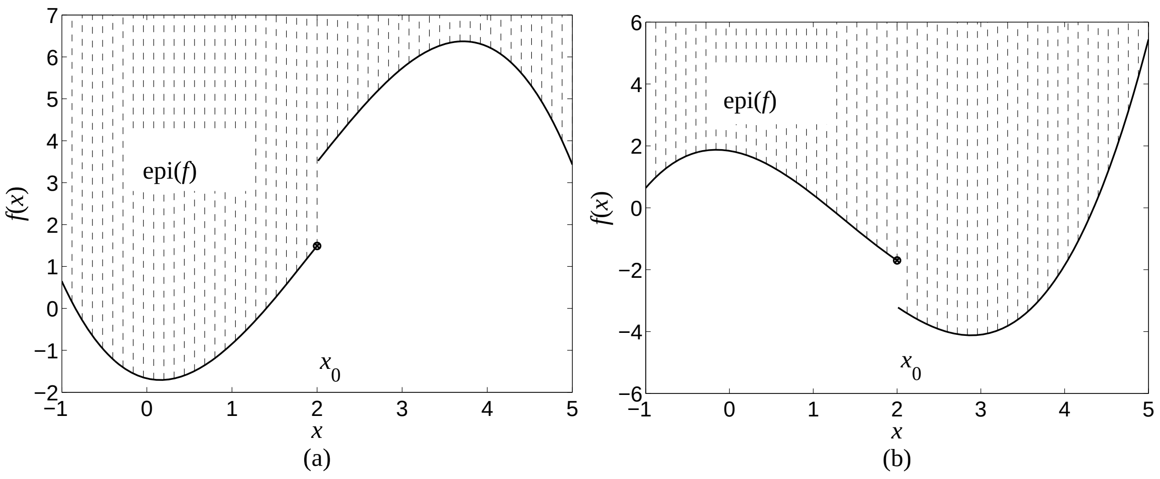


Figure 4.6. (a) the epigraph of a lower-semicontinuous function is a closed set, (b) the epigraph of function that is not lower-semicontinuous is not a closed set.

Definition 4.6 A function $f(\boldsymbol{x})$ is said to belong to class Γ_0 if $f(\boldsymbol{x})$ is a proper, lower-semicontinuous, convex function.

Example 4.4 Let C be a nonempty closed convex set in R^n, the indicator function associated with set C is defined by

$$I_C(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in C \\ +\infty & \text{if } \boldsymbol{x} \notin C \end{cases} \tag{4.20}$$

Verify that $I_C(\boldsymbol{x}) \in \Gamma_0$.

Solution Since the domain $\text{dom}(I_C) = C$ is nonempty, $I_C(\boldsymbol{x})$ is proper. By definition the epigraph of $I_C(\boldsymbol{x})$ is the closed cylinder whose base is the closed convex set C, hence $\text{epi}(I_C)$ is closed and convex. This implies that $I_C(\boldsymbol{x})$ is lower-semicontinuous and convex. Therefore, $I_C(\boldsymbol{x})$ belongs to class Γ_0. ■

Definition 4.7 Let $f(\boldsymbol{x})$ be a proper function, the conjugate of $f(\boldsymbol{x})$ is a function, denoted by $f^*(\boldsymbol{u})$, defined by

$$f^*(\boldsymbol{u}) = \sup_{\boldsymbol{x} \in \text{dom}(f)} \left(\boldsymbol{u}^T \boldsymbol{x} - f(\boldsymbol{x}) \right) \tag{4.21}$$

Given a $\boldsymbol{u} \in R^n$, Eq. (4.21) defines the value $f^*(\boldsymbol{u})$ as the largest difference between the linear function $\boldsymbol{u}^T\boldsymbol{x}$ and function $f(\boldsymbol{x})$. As can be observed from Fig. 4.7, for a smooth $f(\boldsymbol{x})$ this largest difference occurs at a point $\hat{\boldsymbol{x}}$ where the gradient $\nabla f(\boldsymbol{x})$ is equal to $\boldsymbol{u}$. From the figure, it can also be observed that the intersect of the tangent of $f(\boldsymbol{x})$ at $\hat{\boldsymbol{x}}$ is equal to $-f^*(\boldsymbol{u})$.

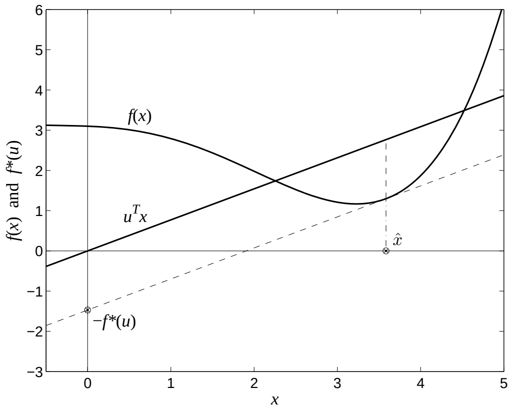


Figure 4.7. Conjugate function $f^*(\boldsymbol{u})$ as the largest difference between $\boldsymbol{u}^T\boldsymbol{x}$ and $f(\boldsymbol{x})$.

Properties of conjugate functions

(a) Conjugate function $f^*(\boldsymbol{u})$ is convex regardless of whether $f(\boldsymbol{x})$ is convex or not.

Proof By definition $f^*(\boldsymbol{u})$ as a function of $\boldsymbol{u}$ is the supremum of a family of affine functions which are convex. Since the supremum of a family of convex functions is convex, $f^*(\boldsymbol{u})$ is convex. ■

(b) For any $\boldsymbol{x}$ and $\boldsymbol{u}$,

$$f(\boldsymbol{x}) + f^*(\boldsymbol{u}) \ge \boldsymbol{x}^T\boldsymbol{u} \tag{4.22}$$

Eq. (4.22) is known as the *Fenchel inequality*.

Proof By Eq. (4.21), we can write

$$f^*(\boldsymbol{u}) = \sup_{\boldsymbol{x} \in \text{dom}(f)} \left(\boldsymbol{u}^T\boldsymbol{x} - f(\boldsymbol{x})\right) \ge \boldsymbol{u}^T\boldsymbol{x} - f(\boldsymbol{x})$$

which leads to Eq. (4.22) immediately. ■

(c) If $f(\boldsymbol{x})$ is a proper and convex function and $\boldsymbol{u} \in \partial f(\boldsymbol{x})$, then

$$f(\boldsymbol{x}) + f^*(\boldsymbol{u}) = \boldsymbol{x}^T\boldsymbol{u} \tag{4.23}$$

Proof The convexity of $f(\boldsymbol{x})$ and $\boldsymbol{u} \in \partial f(\boldsymbol{x})$ imply that

$$f(\boldsymbol{x}_1) \geq f(\boldsymbol{x}) + \boldsymbol{u}^T(\boldsymbol{x}_1 - \boldsymbol{x}) \quad \text{for any} \quad \boldsymbol{x}_1 \in \text{dom}(f)$$

i.e.,

$$\boldsymbol{u}^T\boldsymbol{x} - f(\boldsymbol{x}) \geq \boldsymbol{u}^T\boldsymbol{x}_1 - f(\boldsymbol{x}_1) \quad \text{for any} \quad \boldsymbol{x}_1 \in \text{dom}(f)$$

Hence

$$\boldsymbol{u}^T\boldsymbol{x} - f(\boldsymbol{x}) \geq \sup_{\boldsymbol{x}_1 \in \text{dom}(f)} \left(\boldsymbol{u}^T\boldsymbol{x}_1 - f(\boldsymbol{x}_1)\right)\boldsymbol{x}_1 = f^*(\boldsymbol{u})$$

i.e.,

$$f(\boldsymbol{x}) + f^*(\boldsymbol{u}) \leq \boldsymbol{x}^T\boldsymbol{u}$$

which in conjunction with Eq. (4.22) implies Eq. (4.23). ■

(d) If $f(\boldsymbol{x}) \in \Gamma_0$, then the conjugate of the conjugate function $f^*(\boldsymbol{u})$ is equal to $f(\boldsymbol{x})$. In other words,

$$f^{**}(\boldsymbol{x}) = f(\boldsymbol{x}) \quad \text{for} \quad f(\boldsymbol{x}) \in \Gamma_0 \tag{4.24}$$

Proof By definition

$$f^{**}(\boldsymbol{x}) = \sup_{\boldsymbol{u}}\left(\boldsymbol{x}^T\boldsymbol{u} - f^*(\boldsymbol{u})\right) = \boldsymbol{x}^T\hat{\boldsymbol{u}} - f^*(\hat{\boldsymbol{u}})$$

where $\hat{\boldsymbol{u}}$ achieves the above supremum, and

$$f^*(\hat{\boldsymbol{u}}) = \sup_{\hat{x}}\left(\hat{\boldsymbol{u}}^T\hat{\boldsymbol{x}} - f(\hat{\boldsymbol{x}})\right) \geq \hat{\boldsymbol{u}}^T\boldsymbol{x} - f(\boldsymbol{x})$$

By combining the above two expressions, we obtain

$$f^{**}(\boldsymbol{x}) \leq f(\boldsymbol{x}) \tag{4.25}$$

By applying the Fenchel inequality to $f^*(\boldsymbol{u})$, we obtain

$$f^*(\boldsymbol{u}) + f^{**}(\boldsymbol{x}) \geq \boldsymbol{u}^T\boldsymbol{x}$$

If we take $\boldsymbol{u} \in \partial f(\boldsymbol{x})$ and use Eq. (4.23), the above inequality implies that

$$f^*(\boldsymbol{u}) + f^{**}(\boldsymbol{x}) \geq f(\boldsymbol{x}) + f^*(\boldsymbol{u})$$

i.e., $f^{**}(\boldsymbol{x}) \geq f(\boldsymbol{x})$ which in conjunction with Eq. (4.25) proves Eq. (4.24). ■

(e) If $f(\boldsymbol{x}) \in \Gamma_0$, then $\boldsymbol{u} \in \partial f(\boldsymbol{x})$ if and only if $\boldsymbol{x} \in \partial f^*(\boldsymbol{u})$.

Proof If $\boldsymbol{u} \in \partial f(\boldsymbol{x})$ for a $\boldsymbol{x} \in \text{dom}(f)$, then

$$f(\boldsymbol{x}_1) \ge f(\boldsymbol{x}) + \boldsymbol{u}^T(\boldsymbol{x}_1 - \boldsymbol{x}) \text{ for any } \boldsymbol{x}_1 \in \text{dom}(f)$$

Hence

$$\boldsymbol{u}^T\boldsymbol{x} - f(\boldsymbol{x}) \ge \boldsymbol{u}^T\boldsymbol{x}_1 - f(\boldsymbol{x}_1) \text{ for any } \boldsymbol{x}_1 \in \text{dom}(f)$$

By taking supremum in the above inequality with respect to $\boldsymbol{x}_1$, we obtain $\boldsymbol{u}^T\boldsymbol{x} - f(\boldsymbol{x}) \ge f^*(\boldsymbol{u})$, i.e., $-f(\boldsymbol{x}) \ge f^*(\boldsymbol{u}) - \boldsymbol{u}^T\boldsymbol{x}$. By adding term $\boldsymbol{u}_1^T\boldsymbol{x}$ with arbitrary $\boldsymbol{u}_1$ to both sides of the above inequality, we obtain

$$\boldsymbol{u}_1^T\boldsymbol{x} - f(\boldsymbol{x}) \ge f^*(\boldsymbol{u}) + \boldsymbol{x}^T(\boldsymbol{u}_1 - \boldsymbol{u})$$

Since

$$f^*(\boldsymbol{u}_1) = \sup_{\hat{\boldsymbol{x}}}\left(\boldsymbol{u}_1^T\hat{\boldsymbol{x}} - f(\hat{\boldsymbol{x}})\right) \ge \boldsymbol{u}_1^T\boldsymbol{x} - f(\boldsymbol{x})$$

combining the last two inequalities yields

$$f^*(\boldsymbol{u}_1) \ge f^*(\boldsymbol{u}) + \boldsymbol{x}^T(\boldsymbol{u}_1 - \boldsymbol{u}) \qquad \text{for any } \boldsymbol{u}_1$$

Therefore, $\boldsymbol{x} \in \partial f^*(\boldsymbol{u})$. Conversely, if $\boldsymbol{x} \in \partial f^*(\boldsymbol{u})$, by using a similar argument in conjunction with the property $f^{**}(\boldsymbol{x}) = f(\boldsymbol{x})$, it can be shown that $\boldsymbol{u} \in \partial f(\boldsymbol{x})$. We leave the details of the argument to the reader as an exercise. ■

14.2.5 Proximal operators

Let $f(\boldsymbol{x})$ be a closed, proper, convex function. The proximal operator of $f(\boldsymbol{x})$ is defined by

$$\text{prox}_{\tau f}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x}}\left(f(\boldsymbol{x}) + \tfrac{1}{2\tau}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2\right) \tag{4.26}$$

where $\tau > 0$ is a scaling parameter. With a fixed τ, $\text{prox}_{\tau f}(\boldsymbol{v})$ maps a vector $\boldsymbol{v} \in R^n$ to the unique minimizer of the objective function on the right-hand side of Eq. (4.26). We call $\text{prox}_{\tau f}(\boldsymbol{v})$ an *operator* because obtaining $\text{prox}_{\tau f}(\boldsymbol{v})$ involves a procedure of solving a minimization problem for strongly convex function $f(\boldsymbol{x}) + \frac{1}{2\tau}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2$ rather than straightforward algebraic manipulations of vector $\boldsymbol{v}$. The proximal operator may be regarded as generating points towards the minimizer of $f(\boldsymbol{x})$, and parameter τ controls, relative to point $\boldsymbol{v}$, how much $\text{prox}_{\tau f}(\boldsymbol{v})$ is closer to the minimizer of $f(\boldsymbol{x})$. In effect, under mild assumptions $\text{prox}_{\tau f}(\boldsymbol{v})$ is shown to act like a descent gradient step $\boldsymbol{v} - \tau\nabla f(\boldsymbol{v})$ when τ is small [5].

Properties of proximal operators [5]

(a) Point $\boldsymbol{x}^*$ is a minimizer of $f(\boldsymbol{x})$ if and only if $\boldsymbol{x}^*$ is a fixed point of $\text{prox}_{\tau f}(\boldsymbol{v})$, namely.

$$\boldsymbol{x}^* = \text{prox}_{\tau f}(\boldsymbol{x}^*) \tag{4.27}$$

Proof Suppose $\boldsymbol{x}^*$ is a minimizer of $f(\boldsymbol{x})$, then it minimizes both $f(\boldsymbol{x})$ and $\frac{1}{2\tau}\|\boldsymbol{x}-\boldsymbol{x}^*\|_2^2$, hence it minimizes $f(\boldsymbol{x})+\frac{1}{2\tau}\|\boldsymbol{x}-\boldsymbol{x}^*\|_2^2$, which implies Eq. (4.27). Conversely, if $\boldsymbol{x}^*$ is a fixed point of $\text{prox}_{\tau f}(\boldsymbol{v})$, i.e. Eq. (4.27) holds, then $\boldsymbol{x}^*$ minimizes $f(\boldsymbol{x})+\frac{1}{2\tau}\|\boldsymbol{x}-\boldsymbol{x}^*\|_2^2$. Hence

$$\nabla\left(f(\boldsymbol{x})+\tfrac{1}{2\tau}\|\boldsymbol{x}-\boldsymbol{x}^*\|_2^2\right)\Big|_{\boldsymbol{x}=\boldsymbol{x}^*}=\boldsymbol{0}$$

which implies that $\nabla f(\boldsymbol{x}^*) = 0$. This means that $\boldsymbol{x}^*$ is a minimizer of $f(\boldsymbol{x})$. ■

It can be shown that if $f(\boldsymbol{x})$ possesses a minimizer, then the sequence of iterates produced by

$$\boldsymbol{x}_{k+1}= \text{prox}_{\tau f}(\boldsymbol{x}_k) \qquad \text{for } k = 0, 1, \ldots \tag{4.28}$$

converges to a fixed point $\boldsymbol{x}^*$ of $\text{prox}_{\tau f}(\boldsymbol{v})$ [5], which by the above property is a minimizer of $f(\boldsymbol{x})$. In the literature, algorithms based on Eq. (4.28) are called *proximal-point algorithms*.

(b) For a fully separable function $f(\boldsymbol{x}) = \sum_{i=1}^n f_i(x_i)$,

$$\left(\text{prox}_{\tau f}(\boldsymbol{v})\right)_i = \text{prox}_{\tau f_i}(v_i) \tag{4.29}$$

Proof This is because

$$\begin{aligned}\text{prox}_{\tau f}(\boldsymbol{v}) &= \arg\min_{\boldsymbol{x}}\left(\sum_{i=1}^n f_i(x_i)+\tfrac{1}{2\tau}\sum_{i=1}^n (x_i-v_i)^2\right)\\ &= \arg\min_{\boldsymbol{x}}\left(\sum_{i=1}^n\left(f_i(x_i)+\tfrac{1}{2\tau}(x_i-v_i)^2\right)\right)\end{aligned}$$

hence finding minimizer $\boldsymbol{x}^*$ can be carried out pointwisely by minimizing $f_i(x_i)+\frac{1}{2\tau}(x_i-v_i)^2$ with respect to x_i, i.e., finding

$$\arg\min_{x_i}\left(f_i(x_i)+\tfrac{1}{2\tau}(x_i-v_i)^2\right)=\text{prox}_{\tau f_i}(v_i)$$

This proves Eq. (4.29). ■

(c) The proximal operator can be interpreted as the resolvent of subdifferential operator:

$$\text{prox}_{\tau f} = \left(\boldsymbol{I}+\tau\,\partial f\right)^{-1} \tag{4.30}$$

where $\left(\boldsymbol{I}+\tau\,\partial f\right)^{-1}$ is called the resolvent of subgradient ∂f.

Proof We prove Eq. (4.30) by showing that $\text{prox}_{\tau f}(\boldsymbol{v})=\left(\boldsymbol{I}+\tau\,\partial f\right)^{-1}\boldsymbol{v}$ for arbitrary $\boldsymbol{v}$. Let $\boldsymbol{u}\in\left(\boldsymbol{I}+\tau\,\partial f\right)^{-1}\boldsymbol{v}$. we want to show that $\boldsymbol{u}=\text{prox}_{\tau f}(\boldsymbol{v})$. It follows that $\boldsymbol{v}\in\left(\boldsymbol{I}+\tau\,\partial f\right)\boldsymbol{u}$ = $\boldsymbol{u}+\tau\,\partial f(\boldsymbol{u})$. This implies that $\boldsymbol{0}\in\partial\left(f(\boldsymbol{u})+\frac{1}{2\tau}\|\boldsymbol{u}-\boldsymbol{v}\|_2^2\right)$, hence $\boldsymbol{u}$ minimizes $f(\boldsymbol{x})+\frac{1}{2\tau}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2$, i.e.

$$\boldsymbol{u}=\arg\min_{\boldsymbol{x}}\left(f(\boldsymbol{x})+\tfrac{1}{2\tau}\|\boldsymbol{x}-\boldsymbol{v}\|_2^2\right)=\text{prox}_{\tau f}(\boldsymbol{v})$$

which completes the proof. ■

4.3 Newton Algorithms

This section presents several Newton algorithms [3] that deal with the minimization of a smooth convex function without constraints or subject to constraints which define a convex feasible region.

4.3.1 Minimization of a smooth convex function without constraints

Let $f(\boldsymbol{x})$ be a smooth convex function. Newton's method solves the unconstrained problem

$$\text{minimize}\quad f(\boldsymbol{x})$$

by iteratively minimizing quadratic approximating functions of $f(\boldsymbol{x})$, i.e.,

$$\text{minimize}\ \ f(\boldsymbol{x}_k)+\boldsymbol{g}_k(\boldsymbol{x}-\boldsymbol{x}_k)+\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_k)^T\boldsymbol{H}_k(\boldsymbol{x}-\boldsymbol{x}_k) \tag{4.31}$$

where $\boldsymbol{H}_k=\nabla^2 f(\boldsymbol{x}_k),\ \boldsymbol{g}_k=\nabla f(\boldsymbol{x}_k)$. Because $f(\boldsymbol{x})$ is convex, $\boldsymbol{H}_k$ is positive semidefinite hence is a convex quadratic problem whose solution, $\boldsymbol{x}_{k+1}$, makes the gradient of the quadratic function in Eq. (4.31) zero, namely it satisfies the linear equation

$$\boldsymbol{H}_k(\boldsymbol{x}-\boldsymbol{x}_k)=-\boldsymbol{g}_k$$

The next iterate is therefore given by $\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\alpha_k\boldsymbol{d}_k$ where $\boldsymbol{d}_k$ satisfies the linear equation

$$\boldsymbol{H}_k\boldsymbol{d}_k=-\boldsymbol{g}_k \tag{4.32}$$

and scalar $\alpha_k > 0$ is found by a line search algorithm such as the backtracking search. The Newton algorithm may be terminated if $\|\boldsymbol{x}_{k+1}-\boldsymbol{x}_k\|_2$ is less than a convergence tolerance ε. An alternative and better stopping criteria [3] is to check the Newton decrement as explained below. Let $\boldsymbol{x}^*$ be the minimizer which the Newton iterate $\boldsymbol{x}_k$ converges to, and $f^*=f(\boldsymbol{x}^*)$. Suppose we are in the kth iteration and want to know how far the value of $f(\boldsymbol{x}_k)$ is from the minimum value f^*. Because $\boldsymbol{x}^*$ is not available, f^* is also unknown. Thus we seek to obtain an estimate of $f(\boldsymbol{x}_k) - f^*$ instead. To this end we use the minimum value of the quadratic approximation of $f(\boldsymbol{x})$ at $\boldsymbol{x}_k$ to replace f^*. From Eq. (4.31), we can deduce that

$$f(\boldsymbol{x}_k) - f^* \approx \tfrac{1}{2}\boldsymbol{g}_k^T \boldsymbol{H}_k^{-1} \boldsymbol{g}_k$$

Based on this and Eq. (4.32), the Newton decrement at point $\boldsymbol{x}_k$ is defined by

$$\lambda(\boldsymbol{x}_k) = \left(\boldsymbol{g}_k^T \boldsymbol{H}_k^{-1} \boldsymbol{g}_k\right)^{1/2} = (-\boldsymbol{g}_k^T \boldsymbol{d}_k)^{1/2} \qquad (4.33)$$

Since both $\boldsymbol{g}_k$ and $\boldsymbol{d}_k$ are required in the kth iteration of the algorithm, the additional computation to obtain $\lambda(\boldsymbol{x}_k)$ using Eq. (4.33) is insignificant. The analysis above suggests a stopping criterion based on whether or not $\tfrac{1}{2}\lambda^2(\boldsymbol{x}_k)$ is less than a prescribed tolerance ε. The Newton algorithm for unconstrained minimization of a smooth convex function is summarized below.

Algorithm 4.1 Newton algorithm for unconstrained minimization of convex functions

```
Step 1
Input $\boldsymbol{x}_0$ and initialize the tolerance ε.
Set $k = 0$.
Step 2
Compute $\boldsymbol{g}_k$ and $\boldsymbol{H}_k$.
If the objective function is not strictly convex, modify $\boldsymbol{H}_k$ by adding a scaled identity matrix
$\beta\boldsymbol{I}$ with a small $\beta > 0$.
Compute $\boldsymbol{d}_k$ by solving Eq. (4.32).
Step 3
Compute $\lambda(\boldsymbol{x}_k)$ using Eq. (4.33).
If $\tfrac{1}{2}\lambda^2(\boldsymbol{x}_k) < \varepsilon$, output $\boldsymbol{x}_k$ as the solution and stop. Otherwise, compute $\boldsymbol{d}_k = -\boldsymbol{H}_k^{-1}\boldsymbol{g}_k$.
Step 4
Find $\alpha_k$, the value of α that minimizes $f(\boldsymbol{x}_k + \alpha\,\boldsymbol{d}_k)$, using backtracking line search.
Step 5
Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$.
Set $k = k + 1$ and repeat from Step 2.
```

4.3.2 Minimization of a smooth convex function subject to equality constraints

In this section, we examine constrained CP problems of the form

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to:} \;\; \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{4.34a-b}$$

where $f(\boldsymbol{x})$ is a twice continuously differentiable convex function and $\boldsymbol{A}$ is of full row rank. Eq. (4.34) represents a general class of CP problems with equality constraints because for a problem to be convex, all equality constraints must be linear.

It is well known that point $\boldsymbol{x}$ is a minimizer of the convex problem in Eq. (4.34) if and only if the KKT conditions

$$\boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{A}^T\boldsymbol{\lambda} = \boldsymbol{0}, \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{4.35a-b}$$

are satisfied for some Lagrange multiplier $\boldsymbol{\lambda}$. Now suppose we are in the *k*th iteration with known iterate $\{\boldsymbol{x}_k, \boldsymbol{\lambda}_k\}$ with $\boldsymbol{x}_k$ *primal feasible*, i.e., $\boldsymbol{A}\boldsymbol{x}_k = \boldsymbol{b}$, and we want to update $\{\boldsymbol{x}_k, \boldsymbol{\lambda}_k\}$ to $\{\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}\}$ with $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k$ so as to better satisfy the KKT conditions in Eq. (4.35). To satisfy Eq. (4.35b), namely $\boldsymbol{A}\boldsymbol{x}_{k+1} = \boldsymbol{b}$, we must have

$$\boldsymbol{A}\boldsymbol{d}_k = \boldsymbol{0} \tag{4.36}$$

To deal with Eq. (4.35a), we replace $\boldsymbol{g}(\boldsymbol{x})$ by its first-order Taylor approximation at $\boldsymbol{x}_k$, i.e.,

$$\boldsymbol{g}(\boldsymbol{x}) \approx \boldsymbol{g}_k + \boldsymbol{H}_k\boldsymbol{d}_k$$

so that the KKT condition in Eq. (4.35a) is approximated by

$$\boldsymbol{g}_k + \boldsymbol{H}_k\boldsymbol{d}_k + \boldsymbol{A}^T\boldsymbol{\lambda}_{k+1} = \boldsymbol{0} \tag{4.37}$$

Eqs. (4.36) and (4.37) together gave

$$\begin{bmatrix} \boldsymbol{H}_k & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_k \\ \boldsymbol{\lambda}_{k+1} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{g}_k \\ \boldsymbol{0} \end{bmatrix} \tag{4.38}$$

Note that if no constraints are present Eq. (4.38) will be reduced to Eq. (4.32). For this reason the vector $\boldsymbol{d}_k$ satisfying Eq. (4.38) is regarded as a Newton direction. The solution of Eq. (4.38) is found to be

$$\boldsymbol{d}_k = -\boldsymbol{H}_k^{-1}(\boldsymbol{A}^T\boldsymbol{\lambda}_{k+1} + \boldsymbol{g}_k) \tag{4.39a}$$

where

$$\boldsymbol{\lambda}_{k+1} = -\left(\boldsymbol{A}\boldsymbol{H}_k^{-1}\boldsymbol{A}^T\right)^{-1}\boldsymbol{A}\boldsymbol{H}_k^{-1}\boldsymbol{g}_k \tag{4.39b}$$

Note that the solution $\boldsymbol{d}_k$ obtained from Eq. (4.39) always satisfies Eq. (4.36), hence the second KKT condition in Eq. (4.35b). Consequently, the Newton iteration can safely be terminated as long as the first KKT condition in Eq. (4.35a) is practically satisfied. Based on this, a stopping criterion is deduced as

$$\| \boldsymbol{g}_{k+1} + \boldsymbol{A}^T \boldsymbol{\lambda}_{k+1} \|_2 < \varepsilon \tag{4.40}$$

for a prescribed tolerance ε. The Newton method for the problem in Eq. (4.34) can now be summarized as follows.

Algorithm 4.2 Newton algorithm for convex problems with Equality Constraints

Step 1 Input feasible initial point $\boldsymbol{x}_0 \in \text{dom}(f)$ and a tolerance $\varepsilon > 0$.

Set $k = 0$.

Step 2 Compute $\boldsymbol{d}_k$ and $\boldsymbol{\lambda}_{k+1}$ using Eq. (4.39).

Step **3** Find α_k, the value of α that minimizes $f(\boldsymbol{x}_k + \alpha\, \boldsymbol{d}_k)$, using backtracking line search.

Step 4 Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$.

Compute $\boldsymbol{g}_{k+1}$.

Step 5 If Eq. (4.40) is satisfied, stop and output $\boldsymbol{x}_{k+1}$ as solution; otherwise set $k = k + 1$ and repeat from Step 2.

4.3.3 Newton algorithm for problem in Eq. (4.34) with a nonfeasible $\boldsymbol{x}_0$

Suppose we are in the kth iteration where iterate $\{\boldsymbol{x}_k, \boldsymbol{\lambda}_k\}$ is known but $\boldsymbol{x}_k$ is not necessarily feasible, and we want to update $\{\boldsymbol{x}_k, \boldsymbol{\lambda}_k\}$ to $\{\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}\}$ so as for $\{\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}\}$ to better satisfy the KKT conditions in Eq. (4.35). To satisfy the condition in Eq. (4.35b), we must have

$$\boldsymbol{A}\boldsymbol{d}_k = -(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b})$$

Thus the equation that the Newton direction must satisfy is given by

$$\begin{bmatrix} \boldsymbol{H}_k & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_k \\ \boldsymbol{\lambda}_{k+1} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{g}_k \\ \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k \end{bmatrix} \tag{4.41}$$

Note that $\boldsymbol{d}_k$ satisfying Eq. (4.41) is not necessarily a descent direction because, in case $\boldsymbol{x}_k$ does not satisfy Eq. (4.35b), $\boldsymbol{d}_k$ tends to satisfy the equality constraint in addition to reducing objective function $f(\boldsymbol{x})$. As a result, the line search and stopping criterion used in Algorithm 4.2 is no longer appropriate for the present case. We proceed by letting $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \boldsymbol{\delta}_k$ and writing Eq. (4.41) as

$$\begin{bmatrix} \boldsymbol{H}_k & \boldsymbol{A}^T \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{d}_k \\ \boldsymbol{\delta}_k \end{bmatrix} = -\begin{bmatrix} \boldsymbol{r}_d \\ \boldsymbol{r}_p \end{bmatrix}$$

where $\boldsymbol{r}_d$ and $\boldsymbol{r}_p$ are called *dual* and *primal residuals*, respectively, and are given by

$$\boldsymbol{r}_d = \boldsymbol{g}_k + \boldsymbol{A}^T \boldsymbol{\lambda}_k$$

and

$$\boldsymbol{r}_p = \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b}$$

The solution of the above equation is given by

$$\boldsymbol{d}_k = -\boldsymbol{H}_k^{-1}(\boldsymbol{A}^T \boldsymbol{\delta}_k + \boldsymbol{r}_d) \tag{4.42a}$$

and

$$\boldsymbol{\delta}_k = -\left(\boldsymbol{A}\boldsymbol{H}_k^{-1}\boldsymbol{A}^T\right)^{-1}\left(\boldsymbol{A}\boldsymbol{H}_k^{-1}\boldsymbol{r}_d - \boldsymbol{r}_p\right) \tag{4.42b}$$

It is rational to terminate the algorithm if

$$\| \boldsymbol{r}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) \|_2 < \varepsilon \tag{4.43a}$$

for a prescribed tolerance ε, where

$$\boldsymbol{r}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) \triangleq \begin{bmatrix} \boldsymbol{r}_d \\ \boldsymbol{r}_p \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_k + \boldsymbol{A}^T \boldsymbol{\lambda}_k \\ \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b} \end{bmatrix} \tag{4.43b}$$

To perform line search, we seek to find an $\alpha_k > 0$ that minimizes $\| \boldsymbol{r}(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k, \boldsymbol{\lambda}_k + \alpha \boldsymbol{\delta}_k) \|_2$, where $\{\boldsymbol{d}_k, \boldsymbol{\delta}_k\}$ are given by Eq. (4.42). This one-dimensional minimization can be performed by a backtracking technique as follows.

Algorithm 4.3 Backtracking line search for minimizing $\| \boldsymbol{r}(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k, \boldsymbol{\lambda}_k + \alpha \boldsymbol{\delta}_k) \|_2$

Step 1 Select constants $\rho \in (0, 0.5)$ and $\gamma \in (0, 1)$, say, $\rho = 0.1$ and $\gamma = 0.5$.

Set $\alpha = 1$.

Step 2 If $\| \boldsymbol{r}(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k, \boldsymbol{\lambda}_k + \alpha \boldsymbol{\delta}_k) \|_2 \le (1 - \rho\alpha) \| \boldsymbol{r}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) \|_2$, output $\alpha_k = \alpha$ and stop.

Step 3 Set $\alpha = \gamma\alpha$ and repeat from Step 2.

We now summarize the Newton algorithm as follows.

Algorithm 4.4 Newton algorithm for convex problems with equality constraints and a nonfeasible initial point

Step 1 Input initial point $\boldsymbol{x}_0 \in \mathrm{dom}(f)$, initial $\boldsymbol{\lambda}_0$, and tolerance $\varepsilon > 0$.

Set $k = 0$.

Step 2 If Eq. (4.43a) holds, output $\boldsymbol{x}_k$ as solution and stop.

Step 3 Compute a Newton $\{\boldsymbol{d}_k, \boldsymbol{\delta}_k\}$ by solving Eq. (4.42a).

Step 4 Apply Algorithm 4.3 to find α_k that minimizes $\| \boldsymbol{r}(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k, \boldsymbol{\lambda}_k + \alpha \boldsymbol{\delta}_k) \|_2$.

Step 5 Set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$. Set $k = k + 1$ and repeat from Step 2.

Example 4.5 Apply (a) Algorithm 4.2 and (b) Algorithm 4.4 to the *analytic center* problem

$$\begin{aligned}&\text{minimize } f(\boldsymbol{x}) = -\sum_{i=1}^{n} \log x_i \\ &\text{subject to: } \boldsymbol{e}^T \boldsymbol{x} = \boldsymbol{b}\end{aligned}$$

where $\boldsymbol{e}$ is the all-one column vector of length n, and $b = 1$.

Solution

(a) The gradient and Hessian of $f(\boldsymbol{x})$ are given by $\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} -1/x_1 & -1/x_2 & \cdots & -1/x_n \end{bmatrix}^T$ and $H(\boldsymbol{x}) = diag\{1/x_1^2, 1/x_2^2, \cdots, 1/x_n^2\}$, respectively. By applying Eq. (4.39), we obtain

$$\lambda_{k+1} = \frac{\boldsymbol{e}^T \boldsymbol{x}_k}{\boldsymbol{x}_k^T \boldsymbol{x}_k}, \quad \boldsymbol{d}_k = \boldsymbol{x}_k - \lambda_{k+1} \boldsymbol{x}_k^{(2)}$$

where $\boldsymbol{x}_k^{(2)}$ denotes a vector whose ith component is equal to $\boldsymbol{x}_k^2(i)$. Algorithm 4.2 was evaluated for five instances with n = 10, 50, 100, 200, and 500. For each n, five feasible points were tried. Each of these initial points were generated randomly.

For all 25 cases tested, the convergence tolerance was set to $\varepsilon = 10^{-5}$ and the algorithm was found to converge within 10 iterations. Fig. 4.8 shows a plot of $f(\boldsymbol{x}_k) - f^*$ versus iteration number for the case with n = 200.

(b) To apply Algorithm 4.4, we use Eq. (4.42) to compute the components of $\boldsymbol{d}_k$ and δ_k, which are found to be

$$\begin{aligned}\boldsymbol{d}_k(i) &= -(\delta_k - \boldsymbol{r}_d(i)) \boldsymbol{x}_k^2(i) \\ \delta_k &= -\frac{1}{\boldsymbol{x}_k^T \boldsymbol{x}_k} \left(\sum_{i=1}^{n} \boldsymbol{r}_d(i) \boldsymbol{x}_k^2(i) - r_p \right)\end{aligned}$$

where

$$\boldsymbol{r}_d(i) = \lambda_k - \frac{1}{\boldsymbol{x}_k(i)}, \quad r_p = \boldsymbol{e}^T \boldsymbol{x}_k - 1$$

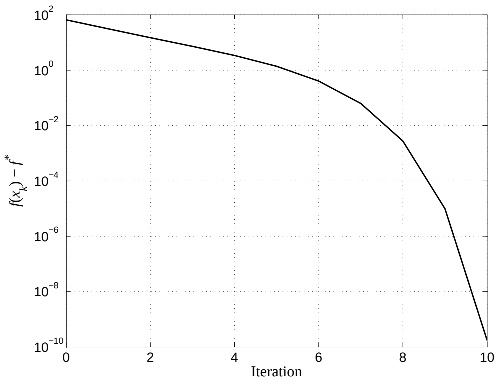


Figure 4.8. $f(\boldsymbol{x}_k) - f^*$ versus iteration number for the analytic center problem with n = 200.

Similar to the set up in part (a), Algorithm 4.4 was evaluated for five instances with n = 10, 50, 100, 200, and 500. For each n, five nonfeasible points were generated randomly with strictly positive entries to ensure that they are in the domain of the objective function. For all 25 cases tested, the convergence tolerance was set to $\varepsilon = 10^{-6}$ and the algorithm was found to converge in less than 25 iterations. Fig. 4.9(a) shows a plot of $\| \boldsymbol{r}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) \|_2$ versus iteration number while Fig. 4.9(b) shows a plot of $f(\boldsymbol{x}_k) - f^*$ versus iteration number for the case of n = 200. We remark that in the implementation of the algorithm a "projection back to dom(*f*)" step was executed in each iteration to project the newly produced iterate $\boldsymbol{x}_k$ back to the domain of the objective function dom(*f*) = $\{\boldsymbol{x}\colon \boldsymbol{x} > \boldsymbol{0}\}$. This was done by verifying the components of $\boldsymbol{x}_k$ and simply setting any component that is less than 10^{-6} to 10^{-6}. The projection step was found helpful to make the algorithm less sensitive to the choice of a nonfeasible initial point. ■

4.3.4 A Newton barrier method for general convex programming problems

We now consider the general convex problem

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) \\ &\text{subject to:} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &&& c_j(x) \le 0 \ \text{ for } j = 1, 2, \ldots, q \end{aligned} \tag{4.44a-c}$$

where $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$ are smooth convex functions. The problem in Eq. (4.44) is equivalent to the problem

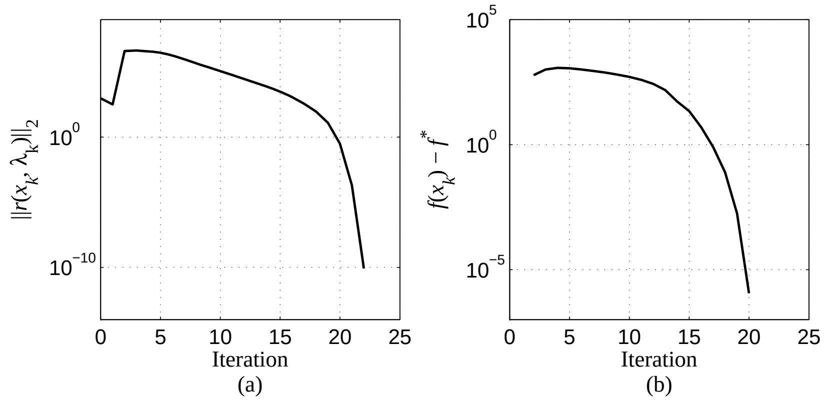


Figure 4.9. (a) residual $\| \boldsymbol{r}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k) \|_2$ and (b) $f(\boldsymbol{x}_k) - f^*$ versus iteration number for the analytic center problem with n = 200. The initial point was not feasible but in dom(f).

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) + \sum_{j=1}^{q} I_-(c_j(\boldsymbol{x})) \\ &\text{subject to:} \ \boldsymbol{Ax} = \boldsymbol{b} \end{aligned} \tag{4.45a-b}$$

where $I_-(u)$ denotes the *indicator function* for nonpositive real u, which is defined by

$$I_-(u) = \begin{cases} 0 & \text{if } u \le 0 \\ \infty & \text{if } u > 0 \end{cases}$$

An effective way to deal with the non-smoothness of the objective function in Eq. (4.45a) is to use a logarithmic function $h_\tau(u)$ $= -(1/\tau)\log(-u)$, which is smooth and convex in its domain u < 0, to approximate $I_-(u)$, where parameter $\tau > 0$ controls approximation error in the sense $\lim_{\tau \to \infty} h_\tau(u) = I_-(u)$ for u < 0 (see Probs. 4.9 and 4.10). With these observations, we proceed by examining the smooth convex problem

$$\begin{aligned} &\text{minimize} \quad F_\tau(\boldsymbol{x}) = f(\boldsymbol{x}) \ - \ \frac{1}{\tau}\sum_{j=1}^{q} \log(-c_j(\boldsymbol{x})) \\ &\text{subject to:} \quad \boldsymbol{Ax} = \boldsymbol{b} \end{aligned} \tag{4.46a-b}$$

which is to be solved in a sequential manner, each with a fixed τ and the value of τ increases as one proceeds. In this regard, the problem in Eq. (4.46) with a fixed τ is fixed is considered as a *subproblem* whose solution is utilized as initial point in solving the next subproblem. The second term of the objective function in Eq. (4.46), namely $b(\boldsymbol{x}) = -\sum_{j=1}^{q} \log(-c_j(\boldsymbol{x}))$, is known as *logarithmic barrier*. Note that $b(\boldsymbol{x})$ is convex as long as all $c_j(\boldsymbol{x})$ are convex. Therefore the problem in Eq. (4.46) is convex. The motivation behind this formulation is that the inclusion of the logarithmic barrier term as a part of its objection function prevents $c_j(\boldsymbol{x})$ from being positive which in turn ensures the constraints in Eq. (4.44c). In the same time, with an increasing τ as the algorithm proceeds, we see a gradually weakening logarithmic barrier relative to function $f(\boldsymbol{x})$. As a result, minimizing $F_\tau(\boldsymbol{x})$ is practically the same as minimizing $f(\boldsymbol{x})$ after a few iterations are carried out. Another formulation equivalent to that in Eq. (4.46) can be obtained by rescaling the objective function as

$$\begin{aligned} &\text{minimize} \quad F_\tau(\boldsymbol{x}) = \tau f(\boldsymbol{x}) + b(\boldsymbol{x}) \\ &\text{subject to:} \quad \boldsymbol{Ax} = \boldsymbol{b} \end{aligned} \tag{4.47a-b}$$

With a fixed τ, the problem in Eq. (4.47) is a smooth convex problem subject to linear equalities, which has been studied in Sec. 4.3.2 where Newton algorithm with feasible initial point is developed. To apply the algorithm, the gradient and Hessian of $F_\tau(\boldsymbol{x})$ are evaluated as $\nabla F_\tau(\boldsymbol{x}) = \tau \nabla f(\boldsymbol{x}) + \nabla b(\boldsymbol{x})$ and $\nabla^2 F_\tau(\boldsymbol{x}) = \tau \nabla^2 f(\boldsymbol{x}) + \nabla^2 b(\boldsymbol{x})$, respectively, where

$$\nabla b(\boldsymbol{x}) = -\sum_{j=1}^{q} \frac{\nabla c_j(\boldsymbol{x})}{c_j(\boldsymbol{x})} \tag{4.48a}$$

$$\nabla^2 b(\boldsymbol{x}) = \sum_{j=1}^{q} \frac{\nabla c_j(\boldsymbol{x}) \nabla^T c_j(\boldsymbol{x})}{c_j(\boldsymbol{x})^2} - \sum_{j=1}^{q} \frac{\nabla^2 c_j(\boldsymbol{x})}{c_j(\boldsymbol{x})} \tag{4.48b}$$

We now address a convergence issue for the Newton barrier method where, for the simplicity of the analysis, the problem in Eq. (4.47) is assumed to have a unique solution $\boldsymbol{x}^*(\tau)$. The KKT conditions of the problem in Eq. (4.47) are given by

$$\begin{gathered} \boldsymbol{Ax}^*(\tau) = \boldsymbol{b}, \quad c_j(\boldsymbol{x}^*(\tau)) < 0 \ \text{ for } \ 1 \le j \le q \\ \tau \nabla f(\boldsymbol{x}^*(\tau)) + \boldsymbol{A}^T \hat{\boldsymbol{\lambda}} + \nabla b(\boldsymbol{x}^*(\tau)) = \boldsymbol{0} \end{gathered}$$

Using Eq. (4.48a), the last equation can be expressed as

$$\nabla f(\boldsymbol{x}^*(\tau)) + \boldsymbol{A}^T \boldsymbol{\lambda}^*(\tau) + \sum_{j=1}^{q} \mu_j^*(\tau) \nabla c_j(\boldsymbol{x}^*(\tau)) = \boldsymbol{0} \tag{4.49a}$$

where

$$\boldsymbol{\lambda}^*(\tau) = \hat{\boldsymbol{\lambda}} / \tau, \quad \mu_j^*(\tau) = -\frac{1}{\tau c_j(\boldsymbol{x}^*(\tau))} \tag{4.49b}$$

Now recall the Lagrangian for the problem in Eq. (4.44) defined by

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T (\boldsymbol{Ax} - \boldsymbol{b}) + \sum_{j=1}^{q} \mu_j c_j(\boldsymbol{x}) \tag{4.50}$$

On comparing Eq. (4.50) with Eq. (4.49), we see that $\boldsymbol{x}^*(\tau)$ minimizes the Lagrangian for $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*(\tau)$, $\mu_j = \mu_j^*(\tau)$ for $1 \le j \le q$, which means that the dual function at $\boldsymbol{\lambda} = \boldsymbol{\lambda}^*(\tau)$ and $\mu_j = \mu_j^*(\tau)$ for $1 \le j \le q$ is given by

$$q(\boldsymbol{\lambda}^*(\tau), \boldsymbol{\mu}^*(\tau)) = f(\boldsymbol{x}^*(\tau)) + \boldsymbol{\lambda}^{*T}(\tau)(\boldsymbol{A}\boldsymbol{x}^*(\tau) - \boldsymbol{b}) + \sum_{j=1}^{q} \mu_j^* c_j(\boldsymbol{x}^*(\tau)) = f(\boldsymbol{x}^*(\tau)) - q/\tau \qquad (4.51)$$

Since the dual function is bounded above by the minimum value f^* of the objective function in the primal problem in Eq. (4.44), Eq. (4.51) implies that

$$f(\boldsymbol{x}^*(\tau)) - f^* \le q/\tau \qquad (4.52)$$

which shows that $\boldsymbol{x}^*(\tau)$ converges to the optimal solution of the problem in Eq. (4.44) as τ goes to infinity. Moreover, Eq. (4.52) also suggests a stopping criterion for the Newton barrier algorithm being q/τ is less than a prescribed tolerance ε. The Newton barrier algorithm can now be outlined as follows.

Algorithm 4.5 Newton barrier algorithm for general convex problem in Eq. (4.44)

Step 1 Input strictly feasible initial point $\boldsymbol{x}_0$, $\tau > 0$ (say, $\tau = 1$), $\gamma > 1$ (say, $\gamma = 10$), and tolerance $\varepsilon > 0$.
Set $k = 0$.

Step 2 Solve the problem in Eq. (4.47) using Algorithm 4.2 to obtain solution $\boldsymbol{x}^*(\tau)$.

Step 3 If $q/\tau < \varepsilon$, stop and output $\boldsymbol{x}^*(\tau)$ as the solution; otherwise set $\tau = \gamma\tau$, $k = k + 1$, and $\boldsymbol{x}_0 = \boldsymbol{x}^*(\tau)$, repeat from Step 2.

We remark that a value of γ in the range [10, 20] usually works well.

The Newton barrier algorithm requires a strictly feasible initial point $\boldsymbol{x}_0$, however finding such a point is not always trivial. Below we describe a method that solves this *feasibility problem* by converting it into a convex problem for which a strictly feasible initial point is easy to find.

The feasibility problem at hand is to find a point $\boldsymbol{x}$ that satisfies $\boldsymbol{Ax} = \boldsymbol{b}$ and $c_j(\boldsymbol{x}) < 0$ for $j = 1, 2, \ldots, q$. First, we find a special solution of $\boldsymbol{Ax} = \boldsymbol{b}$ as $\boldsymbol{x}_s = \boldsymbol{A}^+\boldsymbol{b}$ where $\boldsymbol{A}^+$ is the Moore-Penrose pseudo-inverse of $\boldsymbol{A}$. Next, we find $c_0 = \max\{c_j(\boldsymbol{x}_s), j = 1, 2, \ldots, q\}$. If $c_0 < 0$, then obviously $\boldsymbol{x}_s$ is a strictly feasible point. If $c_0 \ge 0$, then the augmented convex problem

$$\begin{aligned} &\text{minimize} \quad \delta \\ &\text{subject to: } \boldsymbol{Ax} = \boldsymbol{b} \\ &\qquad\qquad c_j(\boldsymbol{x}) \le \delta \ \text{ for } 1 \le j \le q \end{aligned} \tag{4.53}$$

is solved, where δ is treated as an additional variable. In the literature, Eq. (4.53) is referred to as the *phase one optimization problem*. Note that a strictly feasible initial point for the problem in Eq. (4.53) can be identified easily, for example, as $\{\delta_0, \boldsymbol{x}_0\} = \{\hat{c}_0, \boldsymbol{x}_s\}$ where $\hat{c}_0$ is a scalar satisfying $\hat{c}_0 > c_0$. Let the solution of the problem in Eq. (4.53) be denoted by $\{\delta^*, \boldsymbol{x}^*\}$. If $\delta^* < 0$, then $\boldsymbol{x}^*$ can be used as a strictly feasible initial point for the problem in Eq. (4.44). If $\delta^* \ge 0$, then no strictly feasible point exists for the problem.

Example 4.6 Apply Algorithm 4.5 to the LP problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = \boldsymbol{c}^T\boldsymbol{x} \\ &\text{subject to: } \boldsymbol{Ax} \le \boldsymbol{b} \end{aligned} \tag{4.54a-b}$$

where data $\boldsymbol{A} \in R^{100\times 50}$, $\boldsymbol{b} \in R^{100\times 1}$, $\boldsymbol{c} \in R^{50\times 1}$ are randomly generated such that both the primal and dual problems are strictly feasible and the optimal value of $f(\boldsymbol{x})$ is $f^* = 1$.

Solution Denote the *i*th row of $\boldsymbol{A}$ and *i*th entry of $\boldsymbol{b}$ by $\boldsymbol{a}_i^T$ and b_i, respectively. The solution of the problem in Eq. (4.54) is obtained by sequentially solving the subproblem

$$\text{minimize} \quad F_\tau(\boldsymbol{x}) = \tau\boldsymbol{c}^T\boldsymbol{x} - \sum_{i=1}^{100} \log(b_i - \boldsymbol{a}_i^T\boldsymbol{x}) \tag{4.55}$$

We start by finding a strictly feasible initial point $\boldsymbol{x}_0$ such that $\boldsymbol{Ax}_0 - \boldsymbol{b} < 0$. This is done by solving the LP problem

$$\begin{aligned} &\text{minimize} \quad \delta \\ &\text{subject to: } \boldsymbol{Ax} - \boldsymbol{b} \le \delta\boldsymbol{e} \end{aligned}$$

where $\boldsymbol{e}$ denotes the all-one vector of dimension 100. If we denote $\tilde{\boldsymbol{x}} = [\delta \ \ \boldsymbol{x}^T]^T$, it can readily verified that point $\tilde{\boldsymbol{x}}_0 = \begin{bmatrix}\delta_0 & \boldsymbol{0}\end{bmatrix}^T$ with $\delta_0 = \max(-\boldsymbol{b}) + 10^{-3}$ is strictly feasible for the above LP problem. Now suppose the LP problem is solved and the first component of the solution, δ^*, is found negative, then the remaining part of the solution can be taken as a strictly feasible initial point for the problem in Eq. (4.55). We now apply Algorithm 4.5 to the problem in Eq. (4.55) with an initial $\tau = 1$ and three choices of γ: $\gamma = 5$, 10, and 20, respectively. Since the problem in Eq. (4.55) is unconstrained, Step 2 of the algorithm was implemented using Algorithm 4.1 in which the convergence tolerance was set to 10^{-2}, as higher accuracy of the solution for these subproblems are unnecessary. The convergence tolerance ε was set to 10^{-11}. Fig. 4.10

shows the algorithm's performance in terms of $f(\boldsymbol{x}_k) - f^*$ versus cumulative number of Newton steps used. ■

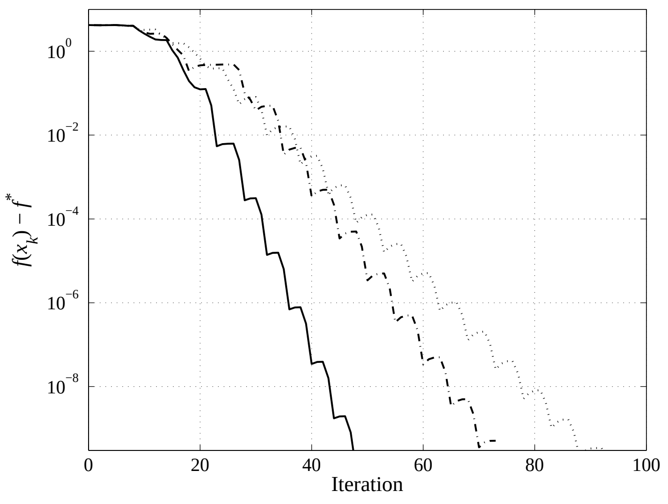


Figure 4.10. Cumulative number of Newton steps versus $f(\boldsymbol{x}_k) - f^*$ for the problem in Example 4.6 where γ was set to 5 (dotted), 10 (dash-dotted), and 20 (solid).

4.4 Minimization of Composite Convex Functions

In this section, we study a class of unconstrained convex problems which assume the form

$$\text{minimize } F(\boldsymbol{x}) = f(\boldsymbol{x}) + \psi(\boldsymbol{x}) \tag{4.56}$$

where function $f(\boldsymbol{x})$ is a differentiable and convex while $\psi(\boldsymbol{x})$ is convex but possibly non-differentiable. The problem in Eq. (4.56) arises from several applications in statistics, engineering, and other fields, and has been a source of motivation in the recent development of theoretical analysis and design of new algorithms for nonsmooth convex optimization in general and minimization of composite convex functions in particular.

4.4.1 A proximal-point algorithm for te problem in Eq. (4.56) [7]

We consider a class of problems that assume the formulation in Eq. (4.56) where $f(\boldsymbol{x})$ is convex with Lipschitz continuous gradient. From (4.9), we can write

$$f(\boldsymbol{x}) + \psi(\boldsymbol{x}) \le f(\boldsymbol{v}) + \nabla f(\boldsymbol{v})^T(\boldsymbol{x} - \boldsymbol{v}) + \psi(\boldsymbol{x}) + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \tag{4.57}$$

for any $\boldsymbol{x}$ and $\boldsymbol{v}$, where L is a Lipschitz constant for the gradient of $f(\boldsymbol{x})$, see Eq. (4.8). If we define

$$\hat{F}(\boldsymbol{x}, \boldsymbol{v}) = f(\boldsymbol{v}) + \nabla f(\boldsymbol{v})^T(\boldsymbol{x} - \boldsymbol{v}) + \psi(\boldsymbol{x}) \tag{4.58}$$

then Eq. (4.57) becomes

$$f(\boldsymbol{x}) + \psi(\boldsymbol{x}) \le \hat{F}(\boldsymbol{x}, \boldsymbol{v}) + \frac{1}{2\tau}\|\boldsymbol{x} - \boldsymbol{v}\|_2^2 \tag{4.59}$$

with $\tau = 1/L$ that induces a proximal-point operator

$$\text{prox}_{\tau\hat{F}}(\boldsymbol{v}) = \arg\min_{\boldsymbol{x}} \left(\hat{F}(\boldsymbol{x}, \boldsymbol{v}) + \tfrac{1}{2\tau} \| \boldsymbol{x} - \boldsymbol{v} \|_2^2 \right) \tag{4.60}$$

Note that the proximal operator defined in Eq. (4.60) differs slightly from that defined in Eq. (4.26) because function $\hat{F}$ depends on both $\boldsymbol{x}$ and $\boldsymbol{v}$. Nevertheless, like a conventional proximal operator, for a given point $\boldsymbol{v}$ $\text{prox}_{\tau\hat{F}}(\boldsymbol{v})$ maps $\boldsymbol{v}$ to a point that is closer to the minimizer of $F(\boldsymbol{x})$ relative to point $\boldsymbol{v}$. This suggests a proximal-point iteration $\boldsymbol{x}_{k+1} = \text{prox}_{\tau\hat{F}}(\boldsymbol{x}_k)$ for k = 0, 1, …, which in conjunction with Eq. (4.60) assumes the form

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \left(\hat{F}(\boldsymbol{x}, \boldsymbol{x}_k) + \tfrac{1}{2\tau} \| \boldsymbol{x} - \boldsymbol{x}_k \|_2^2 \right) \tag{4.61}$$

Theorem 4.4 ***Characterizing minimizer of*** $F(\boldsymbol{x})$ ***in terms of fixed point of a proximal operator*** Point $\boldsymbol{x}^*$ is a minimizer of $F(\boldsymbol{x})$ if and only if $\boldsymbol{x}^*$ is a fixed point of $\text{prox}_{\tau\hat{F}}(\boldsymbol{v})$, that is

$$\boldsymbol{x}^* = \text{prox}_{\tau\hat{F}}(\boldsymbol{x}^*) \tag{4.62}$$

Proof Suppose $\boldsymbol{x}^*$ is a minimizer of $F(\boldsymbol{x})$. From Eq. (14.59) with $\boldsymbol{v} = \boldsymbol{x}^*$, we have

$$f(\boldsymbol{x}) + \psi(\boldsymbol{x}) \le \hat{F}(\boldsymbol{x}, \boldsymbol{x}^*) + \frac{1}{2\tau}\|\boldsymbol{x} - \boldsymbol{x}^*\|_2^2 \tag{4.63}$$

At $\boldsymbol{x} = \boldsymbol{x}^*$, the right-hand side of Eq. (4.63) becomes $f(\boldsymbol{x}^*) + \psi(\boldsymbol{x}^*)$ which is the minimum of the lower bound on the left-hand size of Eq. (4.63). Therefore, $\boldsymbol{x}^*$ minimizes $\hat{F}(\boldsymbol{x}, \boldsymbol{x}^*) + \frac{1}{2\tau}\|\boldsymbol{x} - \boldsymbol{x}^*\|_2^2$ and Eq. (4.62) holds. Conversely, if $\boldsymbol{x}^*$ is a fixed point of $\text{prox}_{\tau\hat{F}}(\boldsymbol{v})$, i.e., Eq. (4.62) holds, then $\boldsymbol{x}^*$ minimizes $\hat{F}(\boldsymbol{x}, \boldsymbol{x}^*) + \frac{1}{2\tau}\|\boldsymbol{x} - \boldsymbol{x}^*\|_2^2$. Hence, by Theorem 4.1, we have

$$\boldsymbol{0} \in \partial \left(\hat{F}(\boldsymbol{x}, \boldsymbol{x}^*) + \tfrac{1}{2\tau} \| \boldsymbol{x} - \boldsymbol{x}^* \|_2^2 \right) \Big|_{\boldsymbol{x}=\boldsymbol{x}^*}$$

which in conjunction with Eq. (4.58) implies that $\boldsymbol{0} \in \nabla f(\boldsymbol{x}^*) + \partial\psi(\boldsymbol{x}^*)$. By Theorem 4.1, this means that $\boldsymbol{x}^*$ is a minimizer of $F(\boldsymbol{x})$. ■

By the definition of $\hat{F}(\boldsymbol{x}, \boldsymbol{x}_k)$, the proximal-point iteration given by Eq. (4.61) can be made more explicit as

$$x_{k+1} = \arg\min_{x} \left(f(x_k) + \nabla f(x_k)^T (x - x_k) + \tfrac{L}{2} \| x - x_k \|_2^2 + \psi(x) \right)$$

which is the same as

$$x_{k+1} = \arg\min_{x} \left(\tfrac{L}{2} \left\| x - \left(x_k - \tfrac{1}{L} \nabla f(x_k) \right) \right\|_2^2 + \psi(x) \right)$$

or

$$x_{k+1} = \arg\min_{x} \left(\tfrac{L}{2} \left\| x - b_k \right\|_2^2 + \psi(x) \right) \tag{4.64a}$$

where

$$b_k = x_k - \tfrac{1}{L} \nabla f(x_k) \tag{4.64b}$$

Algorithm 4.6 Proximal-point algorithm for the problem in Eq. (4.56)

Step 1 Input initial point x_0, Lipschitz constant L for $\nabla f(x)$, and tolerance ε.

Set $k = 0$.

Step 2 Compute x_{k+1} solving the convex problem in Eq. (4.64).

Step 3 If $\| x_{k+1} - x_k \|_2 / \sqrt{n} < \varepsilon$, output solution $x^* = x_{k+1}$ and stop; otherwise

set $k = k + 1$ and repeat from Step 2.

A problem of particular interest arising from several statistics and signal processing applications that fits well into the formulation in Eq. (4.56) is the l_1-l_2 minimization problem

$$\text{minimize} \quad F(x) = \tfrac{1}{2} \|Ax - b\|_2^2 + \mu \| x \|_1 \tag{4.65}$$

where data $A \in R^{m \times n}, b \in R^{m \times 1}$ and *regularization* parameter $\mu > 0$ are given. The two component functions in this case are $f(x) = \tfrac{1}{2} \|Ax - b\|_2^2$ which is convex with Lipschitz continuous gradient and $L = \lambda_{\max}(A^T A)$ (see Example 4.2), and $\Psi(x) = \mu \| x \|_1$ which is convex but non-differentiable. Therefore, the l_1-l_2 minimization problem fits nicely into the formulation in Eq. (4.56). Applying Algorithm 4.6, the iteration step in Eq. (14.64) becomes

$$x_{k+1} = \arg\min_{x} \left(\tfrac{1}{2} \left\| x - b_k \right\|_2^2 + \tfrac{\mu}{L} \| x \|_1 \right) \tag{4.66a}$$

where

$$b_k = \frac{1}{L} A^T \left(b - Ax_k \right) + x_k \tag{4.66b}$$

It is well known [7] that the minimizer of the problem in (4.66) can be obtained by applying the *soft-shrinkage* by μ / L to b_k, i.e.,

$$\boldsymbol{x}_{k+1} = \mathcal{S}_{\mu/L}\left\{\frac{1}{L}\boldsymbol{A}^T\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\right) + \boldsymbol{x}_k\right\} \tag{4.67}$$

where operator $\mathcal{S}_\delta$ is defined by

$$\mathcal{S}_\delta(\boldsymbol{z}) \triangleq \text{sign}(\boldsymbol{z}) \cdot \max\{|\boldsymbol{z}| - \delta, \boldsymbol{0}\} \tag{4.68}$$

where the operations "sign", "max" and "$|\boldsymbol{z}| - \delta$" are performed componentwisely.

Algorithm 4.7 Proximal-point algorithm for the problem in Eq. (4.65)

Step 1 Input data $\{\boldsymbol{A}, \boldsymbol{b}\}$, regularization parameter μ, initial point $\boldsymbol{x}_0$, Lipschitz constant L, and number of iterations K. Set $k = 0$.

Step 2 Compute $\boldsymbol{x}_{k+1}$ using Eq. (4.67).

Step 3 If $k = K$, output solution $\boldsymbol{x}^* = \boldsymbol{x}_{k+1}$ and stop; otherwise set $k = k + 1$ and repeat from Step 2.

4.4.2 A Fast Algorithm for Solving (4.56) [6] [7]

Nesterov [6] presents an algorithm for the problem in Eq. (4.56) with $O(k^{-2})$ rate of convergence in the sense that

$$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{\text{optimal}}) \le \frac{2L\left\|\boldsymbol{x}_0 - \boldsymbol{x}_{\text{optimal}}\right\|_2^2}{(k+1)^2} \tag{4.69}$$

An algorithm, known as *fast iterative shrinkage-thresholding algorithm* (FISTA), is developed in [7], which is essentially a modified version of Nesterov's algorithm, in that soft-thresholding is not directly applied to the preceding iterate but to a subtly modified previous iterate. As such, FISTA is a fast algorithm with $O(k^{-2})$ rate of convergence in the sense of Eq. (4.69). The algorithmic steps of FISTA are outlined as follows.

Algorithm 4.8 FISTA for the problem in Eq. (4.56)

Step 1 Input initial point $\boldsymbol{x}_0$, Lipschitz constant L for $\nabla f(\boldsymbol{x})$, and number of iterations K. Set $\boldsymbol{z}_1 = \boldsymbol{x}_0$, $t_1 = 1$, and $k = 1$.

Step 2 Compute $\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}}\left(\frac{L}{2}\left\|\boldsymbol{x} - \left(\boldsymbol{z}_k - \frac{1}{L}\nabla f(\boldsymbol{z}_k)\right)\right\|_2^2 + \psi(\boldsymbol{x})\right)$

Step 3 Compute $t_{k+1} = \dfrac{1 + \sqrt{1 + 4t_k^2}}{2}$

Step 4 Update $z_{k+1} = x_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x_k - x_{k-1})$

Step 5 If $k = K$, output solution $x^* = x_k$ and stop; otherwise set $k = k + 1$ and repeat from Step 2.

By applying FISTA to the l_1-l_2 minimization problem in Eq. (4.65), we obtain

Algorithm 4.9 FISTA for the problem in Eq. (4.65)

Step 1 Input data $\{A, b\}$, regularization parameter μ, initial point x_0, Lipschitz constant L, and number of iterations K. Set $z_1 = x_0$, $t_1 = 1$, and $k = 1$.

Step 2 Compute $x_k = S_{\mu/L}\left\{\frac{1}{L}A^T(b - Az_k) + z_k\right\}$

Step 3 Compute $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$

Step 4 Update $z_{k+1} = x_k + \left(\frac{t_k - 1}{t_{k+1}}\right)(x_k - x_{k-1})$

Step 5 If $k = K$, output solution $x^* = x_k$ and stop; otherwise set $k = k + 1$ and repeat from Step 2.

We remark that the complexity of Algorithms 4.8 and 4.9 is practically the same as that of Algorithms 4.6 and 4.7, respectively, because the amount of additional computation required by Steps 3 and 4 of FISTA is insignificant.

Example 4.7 ***Deconvolution of distorted and noise-contaminated signals*** Consider a communication channel where the signal received suffers both channel distortion and noise contamination. We assume an FIR channel of length N characterized by transfer function

$$H(z) = \sum_{i=0}^{N-1} h_i z^{-i}$$

The output from the channel is modeled by

$$y(k) = \sum_{i=0}^{N-1} h_i x(k-i) + w(k) \tag{4.70}$$

where $\{w(k)\}$ are additive Gaussian noise with zero mean and variance σ^2. A *deconvolution* problem associated with the model in Eq. (4.70) is to estimate transmitted samples $\{x(k), k = 0, 1, \ldots, n-1\}$ using $n + N - 1$ distorted and noisy measurements $\{y(k), k = 0, 1, \ldots, n + N - 2\}$.

Below we consider a typical case where both *n* and *m* are considerably larger than channel length *N*. Formulate and solve the deconvolution problem as an l_1-l_2 minimization problem.

Solution From Eq. (4.70), we can write

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ y_{n+N-2} \end{bmatrix} = \begin{bmatrix} h_0 & 0 & \cdots & 0 \\ h_1 & h_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ h_{N-1} & h_{N-2} & \cdots & h_0 \\ 0 & h_{N-1} & \cdots & h_1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & h_{N-1} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_{n-1} \end{bmatrix} + \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ w_{n+N-2} \end{bmatrix}$$

i.e.,

$$\boldsymbol{y} = \boldsymbol{Hx} + \boldsymbol{w} \tag{4.71}$$

where $\boldsymbol{H}$ is a *Toeplitz matrix* of size $n + N - 1$ by n determined by its first column and first row.

In practice a signal of interest, like vector $\boldsymbol{x}$ above, is not sparse but compressible under a certain transform such as Fourier or cosine or orthogonal wavelet transform. In analytic terms, this means that there exists an *orthogonal* matrix $\boldsymbol{T}$ of size *n* by *n* that converts a non-sparse $\boldsymbol{x}$ into an approximately sparse $\boldsymbol{\theta}$, namely $\boldsymbol{\theta} = \boldsymbol{Tx}$. Here an approximately sparse signal refers to a discrete signal having a small number of significant nonzero components relative to the signal length, see Fig. 4.11 for an illustrative example.

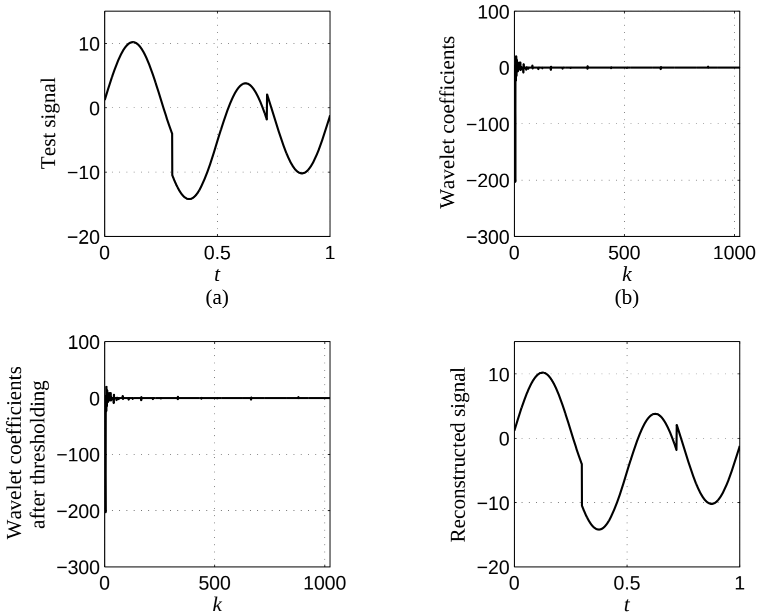


Figure 4.11. (a) A test signal named heavisine of length 1024 (b) Wavelet coefficients of heavisine (c) Wavelet coefficients after hard-thresholding that sets those with magnitude less than 0.05 to zero. This leaves only 10.25% of the coefficients that are nonzero (d) Reconstructed signal using the sparse wavelet coefficients from part (c). The relative reconstruction error in l_2-norm was found to be 3.71×10^{-4}.

With $\boldsymbol{\theta} = \boldsymbol{Tx}$, the model in Eq. (4.71) in terms of $\boldsymbol{\theta}$ becomes

$$\boldsymbol{y} = \boldsymbol{A\theta} + \boldsymbol{w} \tag{4.72}$$

where $\boldsymbol{A} = \boldsymbol{H}\boldsymbol{T}^T$, and the deconvolution problem can be tackled by solving the convex problem

$$\begin{aligned} &\text{minimize} \quad \|\boldsymbol{\theta}\|_1 \\ &\text{subject to:} \ \|\boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{y}\|_2 \le \varepsilon \end{aligned} \tag{4.73}$$

The bound ε in Eq. (4.73) is related to channel noise $\boldsymbol{w}$ and can be estimated using Eq. (4.72) as

$$\varepsilon = \|\boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{y}\|_2 = \|\boldsymbol{w}\|_2 \approx \sqrt{n+N-1}\,\sigma \tag{4.74}$$

With an appropriate weight $\mu > 0$, the problem in Eq. (4.73) is equivalent to the l_1-l_2 problem

$$\text{minimize} \ F(\boldsymbol{\theta}) = \tfrac{1}{2}\|\boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{y}\|_2^2 + \mu\|\boldsymbol{\theta}\|_1 \tag{4.75}$$

As an example we apply Algorithm 4.7 to problem (4.75), where signal $\boldsymbol{x}$ is heavisine of length 1024 (see Fig. 4.12a), the FIR channel is characterized by a narrow-band lowpass filter of length $N = 37$ with a passband of $[0, 0.1\pi]$, and $\boldsymbol{w}$ is a noise vector with components that are Gaussian white with mean 0 and variance $\sigma^2 = 0.25$. As a result, $\boldsymbol{H}$ is a *Toeplitz matrix* of size 1060 by 1024. With $\mu = 0.245$, it took the algorithm 165 iterations to converge to a solution $\boldsymbol{\theta}^*$, which was used to construct signal $\boldsymbol{x}^* = \boldsymbol{T}^T\boldsymbol{\theta}^*$. The signal-to-noise ratio (SNR) of the deconvolved signal was found to be 27.2769 dB. For comparison, the SNR of the distorted and noise corrupted signal $\boldsymbol{y}$ was 5.7592 dB. The original, distorted and recovered signals are depicted in Fig. 4.12.

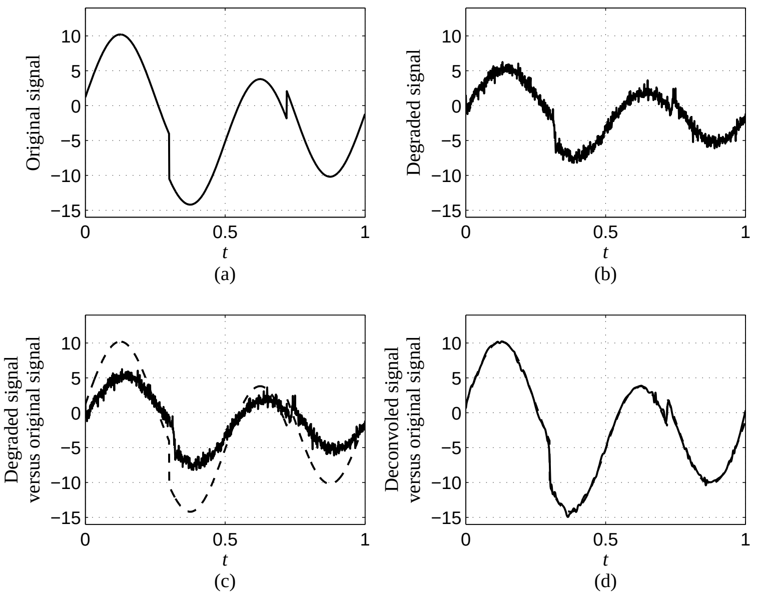


Figure 4.12. (a) Original signal heavisine (b) Distorted and noise-contaminated heavisine (c) Original (dashed) versus degraded (solid) heavisine, SNR = 5.7592 dB (d) Original (dashed) versus reconstructed (solid) heavisine, SNR = 27.2769 dB.

Algorithm 4.9 was also applied to the same deconvoluation problem. With $\mu = 0.245$, it took the algorithm 35 iterations to converge to a solution $\boldsymbol{\theta}^*$. The SNR of the deconvolved signal was found to be 27.2745 dB which is practically the same as that obtained by Algorithm 4.7. As expected, Algorithm 4.9 was considerably faster: the average CPU time it required was approximately 20% of that required by Algorithm 4.7. ■

4.5 Alternating Direction Methods

Alternating direction methods have become increasingly important because of their ability to deal with large scale convex problems. This section presents two representative classes of alternating direction methods known as *alternating direction methods of multipliers* and *alternating minimization algorithms*.

4.5.1 Alternating direction method of multipliers

The alternating direction methods of multipliers (ADMM) [8] are aimed at solving the class of convex problems

$$\begin{aligned} &\text{minimize } \ f(\boldsymbol{x}) + h(\boldsymbol{y}) \\ &\text{subject to: } \boldsymbol{Ax} + \boldsymbol{By} = \boldsymbol{c} \end{aligned} \tag{4.76a-b}$$

where $\boldsymbol{x} \in R^n$ and $\boldsymbol{y} \in R^m$ are variables, $\boldsymbol{A} \in R^{p\times n}$, $\boldsymbol{B} \in R^{p\times m}$, $\boldsymbol{c} \in R^{p\times 1}$, and $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ are convex functions. Note that in Eq. (4.76) the variables in both objective function and constraint are split into two parts, each involves only one set of variables. By definition, the Lagrangian for the problem in Eq. (4.76) is given by

$$L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + h(\boldsymbol{y}) + \boldsymbol{\lambda}^T(\boldsymbol{Ax} + \boldsymbol{By} - \boldsymbol{c})$$

If both $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ are differentiable, the KKT conditions for the problem in Eq. (4.76) are given by

$$\begin{aligned} &\boldsymbol{Ax} + \boldsymbol{By} = \boldsymbol{c} \\ &\nabla f(\boldsymbol{x}) + \boldsymbol{A}^T\boldsymbol{\lambda} = \boldsymbol{0} \\ &\nabla h(\boldsymbol{y}) + \boldsymbol{B}^T\boldsymbol{\lambda} = \boldsymbol{0} \end{aligned} \tag{4.77a-c}$$

The Lagrange dual of Eq. (4.76) assumes the form

$$\text{maximize } \ q(\boldsymbol{\lambda}) \tag{4.78}$$

where

$$q(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x},\boldsymbol{y}} \left[f(\boldsymbol{x}) + h(\boldsymbol{y}) + \boldsymbol{\lambda}^T(\boldsymbol{Ax} + \boldsymbol{By} - \boldsymbol{c}) \right]$$

which in conjunction with Eq. (4.21) leads to

$$\begin{aligned} q(\boldsymbol{\lambda}) &= \inf_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \boldsymbol{\lambda}^T\boldsymbol{Ax} \right] + \inf_{\boldsymbol{y}} \left[h(\boldsymbol{y}) + \boldsymbol{\lambda}^T\boldsymbol{By} \right] - \boldsymbol{\lambda}^T\boldsymbol{c} \\ &= -\sup_{\boldsymbol{x}} \left[(-\boldsymbol{A}^T\boldsymbol{\lambda})^T\boldsymbol{x} - f(\boldsymbol{x}) \right] - \sup_{\boldsymbol{y}} \left[(-\boldsymbol{B}^T\boldsymbol{\lambda})^T\boldsymbol{y} - h(\boldsymbol{y}) \right] - \boldsymbol{\lambda}^T\boldsymbol{c} \end{aligned} \tag{4.79}$$

$$= -f^*(-\boldsymbol{A}^T\boldsymbol{\lambda}) - h^*(-\boldsymbol{B}^T\boldsymbol{\lambda}) - \boldsymbol{c}^T\boldsymbol{\lambda}$$

By the properties that $\boldsymbol{u} = \nabla f(\boldsymbol{v})$ if and only if $\boldsymbol{v} = \nabla f^*(\boldsymbol{u})$ and that $\nabla f^*(-\boldsymbol{A}^T\boldsymbol{\lambda}) = -\boldsymbol{A}\nabla f^*(\boldsymbol{\lambda})$ (see Sec. 4.2.4), Eq. (4.79) implies that

$$\nabla q(\boldsymbol{\lambda}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \tag{4.80}$$

where $\{\boldsymbol{x}, \boldsymbol{y}\}$ minimizes $L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda})$ for a given $\boldsymbol{\lambda}$.

If in addition we assume that $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ are strictly convex, a solution of the problem in Eq. (4.76) can be found by minimizing the Lagranging $L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}^*)$ with respect to primal variables $\boldsymbol{x}$ and $\boldsymbol{y}$, where $\boldsymbol{\lambda}^*$ maximizes the dual function $q(\boldsymbol{\lambda})$ in Eq. (4.79). This in conjunction with Eq. (4.80) suggests *dual ascent* iterations for the problem in Eq. (4.76) as follows:

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{y}_k, \boldsymbol{\lambda}_k) = \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \boldsymbol{\lambda}_k^T \boldsymbol{A}\boldsymbol{x} \right] \\
\boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}} L(\boldsymbol{x}_k, \boldsymbol{y}, \boldsymbol{\lambda}_k) = \arg\min_{\boldsymbol{y}} \left[h(\boldsymbol{y}) + \boldsymbol{\lambda}_k^T \boldsymbol{B}\boldsymbol{y} \right] \\
\boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \alpha_k (\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y}_{k+1} - \boldsymbol{c})
\end{aligned} \tag{4.81a-c}$$

where scalar $\alpha_k > 0$ is chosen to maximize $q(\boldsymbol{\lambda})$ (see Eq. (4.78)) along the direction $\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y}_{k+1} - \boldsymbol{c}$.

It is well known that convex problems of form in Eq. (4.76) with less restrictive $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ and data matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be handled by augmented dual based on the *augmented Lagrangian* [8]

$$L_\alpha(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + h(\boldsymbol{y}) + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c}) + \tfrac{\alpha}{2} \| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \|_2^2 \tag{4.82}$$

which includes the conventional Lagrangian $L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda})$ as a special case when parameter α is set to zero. The introduction of augmented Lagrangian may be understood by considering the following [8]: if we modify the objective function in Eq. (4.76) by adding a penalty term $\tfrac{\alpha}{2} \| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \|_2^2$ for violation of the equality constraint, namely,

$$\begin{aligned}
&\text{minimize} \quad f(\boldsymbol{x}) + h(\boldsymbol{y}) + \tfrac{\alpha}{2} \| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \|_2^2 \\
&\text{subject to: } \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} = \boldsymbol{c}
\end{aligned} \tag{4.83}$$

then the conventional Lagrangian of problem (4.83) is exactly equal to $L_\alpha(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda})$ in Eq. (4.82).

Associated with $L_\alpha(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda})$, the augmented dual problem is given by

$$\text{maximize} \quad q_\alpha(\boldsymbol{\lambda})$$

where

$$q_\alpha(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x},\boldsymbol{y}} \left[f(\boldsymbol{x}) + h(\boldsymbol{y}) + \boldsymbol{\lambda}^T(\boldsymbol{Ax}+\boldsymbol{By}-\boldsymbol{c}) + \tfrac{\alpha}{2} \| \boldsymbol{Ax}+\boldsymbol{By}-\boldsymbol{c} \|_2^2 \right]$$

Unlike the dual ascent iterations in Eq. (4.81) where the minimization of the Lagrangian with respect to variables {***x***, ***y***} is split into two separate steps with reduced problem size, the augmented Lagrangian are no longer separable in variables ***x*** and ***y*** because of the presence of the penalty term. In ADMM iterations, this issue is addressed by *alternating* updates of the primal variables ***x*** and ***y***, namely,

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \boldsymbol{\lambda}_k^T \boldsymbol{Ax} + \tfrac{\alpha}{2} \| \boldsymbol{Ax}+\boldsymbol{By}_k-\boldsymbol{c} \|_2^2 \right] \\
\boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}} \left[h(\boldsymbol{y}) + \boldsymbol{\lambda}_k^T \boldsymbol{By} + \tfrac{\alpha}{2} \| \boldsymbol{Ax}_{k+1}+\boldsymbol{By}-\boldsymbol{c} \|_2^2 \right] \qquad (4.84\text{a-c}) \\
\boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \alpha(\boldsymbol{Ax}_{k+1}+\boldsymbol{By}_{k+1}-\boldsymbol{c})
\end{aligned}$$

Note that parameter α from the quadratic penalty term is used in Eq. (4.84c) to update Lagrange multiplier $\boldsymbol{\lambda}_k$, thereby eliminating a line search step to compute α_k as required in Eq. (4.81c). To justify Eq. (4.84), note that $\boldsymbol{y}_{k+1}$ minimizes $h(\boldsymbol{y}) + \boldsymbol{\lambda}_k^T \boldsymbol{By} + \tfrac{\alpha}{2} \| \boldsymbol{Ax}_{k+1}+\boldsymbol{By}-\boldsymbol{c} \|_2^2$, hence

$$\begin{aligned}
\boldsymbol{0} &= \nabla h(\boldsymbol{y}_{k+1}) + \boldsymbol{B}^T\boldsymbol{\lambda}_k + \alpha \boldsymbol{B}^T(\boldsymbol{Ax}_{k+1}+\boldsymbol{By}_{k+1}-\boldsymbol{c}) \\
&= \nabla h(\boldsymbol{y}_{k+1}) + \boldsymbol{B}^T\left[\boldsymbol{\lambda}_k + \alpha(\boldsymbol{Ax}_{k+1}+\boldsymbol{By}_{k+1}-\boldsymbol{c})\right]
\end{aligned}$$

which in conjunction with Eq. (4.84c) leads to

$$\nabla h(\boldsymbol{y}_{k+1}) + \boldsymbol{B}^T\boldsymbol{\lambda}_{k+1} = 0$$

Therefore, the KKT condition in Eq. (4.77c) is satisfied by ADMM iterations. In addition, since $\boldsymbol{x}_{k+1}$ minimizes $f(\boldsymbol{x}) + \boldsymbol{\lambda}_k^T \boldsymbol{Ax} + \tfrac{\alpha}{2} \| \boldsymbol{Ax}+\boldsymbol{By}_k-\boldsymbol{c} \|_2^2$, we have

$$\begin{aligned}
\boldsymbol{0} &= \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T\boldsymbol{\lambda}_k + \alpha \boldsymbol{A}^T(\boldsymbol{Ax}_{k+1}+\boldsymbol{By}_k-\boldsymbol{c}) \\
&= \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T\left[\boldsymbol{\lambda}_k + \alpha(\boldsymbol{Ax}_{k+1}+\boldsymbol{By}_k-\boldsymbol{c})\right] \\
&= \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T\boldsymbol{\lambda}_{k+1} - \alpha \boldsymbol{A}^T\boldsymbol{B}(\boldsymbol{y}_{k+1}-\boldsymbol{y}_k)
\end{aligned}$$

i.e.,

$$\nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T\boldsymbol{\lambda}_{k+1} = \alpha \boldsymbol{A}^T\boldsymbol{B}(\boldsymbol{y}_{k+1}-\boldsymbol{y}_k) \qquad (4.85)$$

On comparing Eq. (4.85) with Eq. (4.77b), a *dual residual* in the *k*th iteration can be defined as

$$\boldsymbol{d}_k = \alpha \boldsymbol{A}^T\boldsymbol{B}(\boldsymbol{y}_{k+1}-\boldsymbol{y}_k) \qquad (4.86)$$

From (4.77a), a *primal residual* in the *k*th iteration is defined as

$$\boldsymbol{r}_k = \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y}_{k+1} - \boldsymbol{c} \tag{4.87}$$

Together, $\{\boldsymbol{r}_k, \boldsymbol{d}_k\}$ measures closeness of the *k*th ADMM iteration $\{\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{\lambda}_k\}$ to the solution of the problem in Eq. (4.76), thus a reasonable criteria for terminating ADMM iterations is when

$$\| \boldsymbol{r}_k \| < \varepsilon_p \text{ and } \|\boldsymbol{d}_k \| < \varepsilon_d \tag{4.88}$$

where ε_p and ε_d are prescribed tolerances for primal and dual residuals, respectively.

Convergence of the ADMM iterations in Eq. (4.84) have been investigated under various assumptions, see [8] and [9] and the references cited therein. If both $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ are strongly convex with parameters m_f and m_h, respectively, and parameter α is chosen to satisfy

$$\alpha^3 \le \frac{m_f m_h^2}{\rho(\boldsymbol{A}^T\boldsymbol{A})\rho(\boldsymbol{B}^T\boldsymbol{B})^2}$$

where $\rho(\boldsymbol{M})$ denotes the largest eigenvalue of symmetric matrix $\boldsymbol{M}$, then both primal and dual residuals vanish at rate $O(1/k)$ [9], namely,

$$\| \boldsymbol{r}_k \|_2 \le O(1/k) \text{ and } \| \boldsymbol{d}_k \|_2 \le O(1/k)$$

We now summarize the method for solving the problem in Eq. (4.76) as an algorithm.

Algorithm 4.10 ADMM for the problem in Eq. (4.76)

Step 1 Input parameter $\alpha > 0$, $\boldsymbol{y}_0$, $\boldsymbol{\lambda}_0$, and tolerance $\varepsilon_p > 0$, $\varepsilon_d > 0$.

Set $k = 0$.

Step 2 Compute $\{ \{\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}, \boldsymbol{\lambda}_{k+1}\}$ using Eq. (4.84).

Step 3 Compute $\boldsymbol{d}_k$ and $\boldsymbol{r}_k$ using Eqs. (4.86) and (4.87), respectively.

Step 4 If the conditions in Eq. (4. 88) are satisfied, output $(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1})$ as solution and stop; Otherwise, set $k = k + 1$ and repeat from Step 2.

Several variants of ADMM are available, one of them is that of the *scaled form* ADMM [8]. By letting

$$\boldsymbol{r} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \text{ and } \boldsymbol{v} = \boldsymbol{\lambda} / \alpha,$$

we write the augmented Lagrangian as

$$\begin{aligned}
L_\alpha(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda}) &= f(\boldsymbol{x})+h(\boldsymbol{y})+\boldsymbol{\lambda}^T\boldsymbol{r}+\tfrac{\alpha}{2}\|\boldsymbol{r}\|_2^2\\
&= f(\boldsymbol{x})+h(\boldsymbol{y})+\tfrac{\alpha}{2}\|\boldsymbol{r}+\boldsymbol{v}\|_2^2-\tfrac{\alpha}{2}\|\boldsymbol{v}\|_2^2\\
&= f(\boldsymbol{x})+h(\boldsymbol{y})+\tfrac{\alpha}{2}\|\boldsymbol{Ax}+\boldsymbol{By}-\boldsymbol{c}+\boldsymbol{v}\|_2^2-\tfrac{\alpha}{2}\|\boldsymbol{v}\|_2^2
\end{aligned}$$

Consequently, the scaled ADMM algorithm can be outlined as follows.

Algorithm 4.11 Scaled ADMM for the problem in Eq. (4.76)

Step 1 Input parameter $\alpha > 0$, $\boldsymbol{y}_0$, $\boldsymbol{v}_0$, and tolerance $\varepsilon_p > 0$, $\varepsilon_d > 0$.

Set $k = 0$.

Step 2 Compute

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}}\left[f(\boldsymbol{x})+\tfrac{\alpha}{2}\|\boldsymbol{Ax}+\boldsymbol{By}_k-\boldsymbol{c}+\boldsymbol{v}_k\|_2^2\right]\\
\boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}}\left[h(\boldsymbol{y})+\tfrac{\alpha}{2}\|\boldsymbol{Ax}_{k+1}+\boldsymbol{By}-\boldsymbol{c}+\boldsymbol{v}_k\|_2^2\right]\\
\boldsymbol{v}_{k+1} &= \boldsymbol{v}_k+\boldsymbol{Ax}_{k+1}+\boldsymbol{By}_{k+1}-\boldsymbol{c}
\end{aligned}\tag{4.89}$$

Step 3 Compute $\boldsymbol{d}_k$ and $\boldsymbol{r}_k$ using Eqs. (4.86) and (4.87), respectively.

Step 4 If the conditions in Eq. (4. 88) are satisfied, output ($\boldsymbol{x}_{k+1}$, $\boldsymbol{y}_{k+1}$) as solution and stop; Otherwise, set $k = k + 1$ and repeat from Step 2.

A variant of scaled ADMM is the *over-relaxed* ADMM in that the term $\boldsymbol{Ax}_{k+1}$ in the $\boldsymbol{y}$- and $\boldsymbol{v}$-updates is replaced by $\tau\boldsymbol{Ax}_{k+1}-(1-\tau)(\boldsymbol{By}_k-\boldsymbol{c})$ with $\tau\in(0,\,2]$. Thus the over-relaxed ADMM iterations assume the form

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}}\left[f(\boldsymbol{x})+\tfrac{\alpha}{2}\|\boldsymbol{Ax}+\boldsymbol{By}_k-\boldsymbol{c}+\boldsymbol{v}_k\|_2^2\right]\\
\boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}}\left[h(\boldsymbol{y})+\tfrac{\alpha}{2}\|\tau\boldsymbol{Ax}_{k+1}-(1-\tau)\boldsymbol{By}_k+\boldsymbol{By}-\tau\boldsymbol{c}+\boldsymbol{v}_k\|_2^2\right]\\
\boldsymbol{v}_{k+1} &= \boldsymbol{v}_k+\tau\boldsymbol{Ax}_{k+1}-(1-\tau)\boldsymbol{By}_k+\boldsymbol{By}_{k+1}-\tau\boldsymbol{c}
\end{aligned}\tag{4.90}$$

In addition, the idea from Nesterov's accelerated gradient descent algorithm [2] has been extended to ADMM [9]. If both functions $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ are strongly convex, then a fast ADMM outlined below is shown to converge [9].

Algorithm 4.12 Accelerated ADMM for the problem in Eq. (4.76)

Step 1 Input parameter $\alpha > 0$, $\hat{\boldsymbol{y}}_0$, $\hat{\boldsymbol{\lambda}}_0$, and tolerance $\varepsilon_p > 0$, $\varepsilon_d > 0$.

Set $\boldsymbol{y}_{-1} = \hat{\boldsymbol{y}}_0, \boldsymbol{\lambda}_{-1} = \hat{\boldsymbol{\lambda}}_0$, $t_0 = 1$, and $k = 0$.

Step 2 Compute

$$\begin{aligned}
\boldsymbol{x}_k &= \arg\min_{\boldsymbol{x}}\left[f(\boldsymbol{x}) + \hat{\boldsymbol{\lambda}}_k^T \boldsymbol{A}\boldsymbol{x} + \tfrac{\alpha}{2}\| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\hat{\boldsymbol{y}}_k - \boldsymbol{c} \|_2^2 \right]\\
\boldsymbol{y}_k &= \arg\min_{\boldsymbol{y}}\left[h(\boldsymbol{y}) + \hat{\boldsymbol{\lambda}}_k^T \boldsymbol{B}\boldsymbol{y} + \tfrac{\alpha}{2}\| \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \|_2^2 \right]\\
\boldsymbol{\lambda}_k &= \hat{\boldsymbol{\lambda}}_k + \alpha(\boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{y}_k - \boldsymbol{c})\\
t_{k+1} &= \tfrac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)\\
\hat{\boldsymbol{y}}_{k+1} &= \boldsymbol{y}_k + \tfrac{t_k - 1}{t_{k+1}}(\boldsymbol{y}_k - \boldsymbol{y}_{k-1})\\
\hat{\boldsymbol{\lambda}}_{k+1} &= \boldsymbol{\lambda}_k + \tfrac{t_k - 1}{t_{k+1}}(\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1})
\end{aligned} \tag{4.91}$$

Step 3 If $\| \alpha \boldsymbol{A}^T \boldsymbol{B}(\boldsymbol{y}_k - \boldsymbol{y}_{k-1}) \|_2 < \varepsilon_d$ and $\| \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{y}_k - \boldsymbol{c} \|_2 < \varepsilon_p$, output ($\boldsymbol{x}_k$, $\boldsymbol{y}_k$) as solution and stop; Otherwise, set $k = k + 1$ and repeat from Step 2.

Example 4.8

(a) Apply Algorithm 4.10 to solve the l_1-l_2 minimization problem

$$\text{minimize} \quad \tfrac{1}{2}\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \|_2^2 + \mu \| \boldsymbol{x} \|_1 \tag{4.92}$$

(b) Apply the results from part (a) to solve the deconvolution problem in Example 4.7.

Solution (a) The problem in Eq. (4.92) can be formulated as [8]

$$\begin{aligned}
&\text{minimize} \quad \tfrac{1}{2}\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \|_2^2 + \mu \| \boldsymbol{y} \|_1\\
&\text{subject to:} \quad \boldsymbol{x} - \boldsymbol{y} = \boldsymbol{0}
\end{aligned}$$

which fits into the formulation in Eq. (4.76) with $f(\boldsymbol{x}) = \tfrac{1}{2}\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \|_2^2$ and $h(\boldsymbol{y}) = \mu \| \boldsymbol{y} \|_1$. The scaled ADMM iterations in this case assume the form

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}}\left[\tfrac{1}{2}\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \|_2^2 + \tfrac{\alpha}{2}\| \boldsymbol{x} - (\boldsymbol{y}_k - \boldsymbol{\lambda}_k / \alpha) \|_2^2 \right]\\
\boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}}\left[\mu \| \boldsymbol{y} \|_1 + \tfrac{\alpha}{2}\| \boldsymbol{y} - (\boldsymbol{x}_{k+1} + \boldsymbol{\lambda}_k / \alpha) \|_2^2 \right]\\
\boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \alpha(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1})
\end{aligned}$$

Evidently, updating $\boldsymbol{x}_k$ amounts to minimizing a convex quadratic function and updating $\boldsymbol{y}_k$ can be done by soft-shrinkage of $\boldsymbol{x}_{k+1} + \boldsymbol{\lambda}_k / \alpha$ by μ / α, see Eq. (4.67). Thus we have

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= (\boldsymbol{A}^T \boldsymbol{A} + \alpha \boldsymbol{I})^{-1}(\boldsymbol{A}^T \boldsymbol{b} + \alpha \boldsymbol{y}_k - \boldsymbol{\lambda}_k)\\
\boldsymbol{y}_{k+1} &= \mathcal{S}_{\mu/\alpha}(\boldsymbol{x}_{k+1} + \boldsymbol{\lambda}_k / \alpha)\\
\boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \alpha(\boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1})
\end{aligned} \tag{4.93}$$

where operator $\mathcal{S}_{\mu/\alpha}$ is defined by Eq. (4.68). Note that matrix $(\boldsymbol{A}^T \boldsymbol{A} + \alpha \boldsymbol{I})^{-1}$ as well as vector $\boldsymbol{A}^T \boldsymbol{b}$ in Eq. (4.93) are independent of iterations, hence they need to be computed only once.

(b) By applying Eq. (4.93) to the data set described in Example 4.7 with $\boldsymbol{y}_0 = \boldsymbol{0}$, $\boldsymbol{\lambda}_0 = \boldsymbol{0}$, $\mu = 0.25$, and $\alpha = 0.11$, it took 32 ADMM iterations to yield a satisfactory estimation of the signal. The SNR of the estimated signal was found to be 27.2765 dB. The original, distorted, and recovered signals are depicted in Fig. 4.13. The profiles of the primal and dual residuals in terms of $\|\boldsymbol{r}_k\|_2$ and $\|\boldsymbol{d}_k\|_2$ are shown in Fig. 4.14. It is observed that both $\|\boldsymbol{r}_k\|_2$ and $\|\boldsymbol{d}_k\|_2$ fall below 5×10^{-3} after 32 iterations. On comparing with the proximal-point algorithms in Sec. 4.4, the average CPU time required by ADMM iterations was found be practically the same as that of Algorithm 14.9. ■

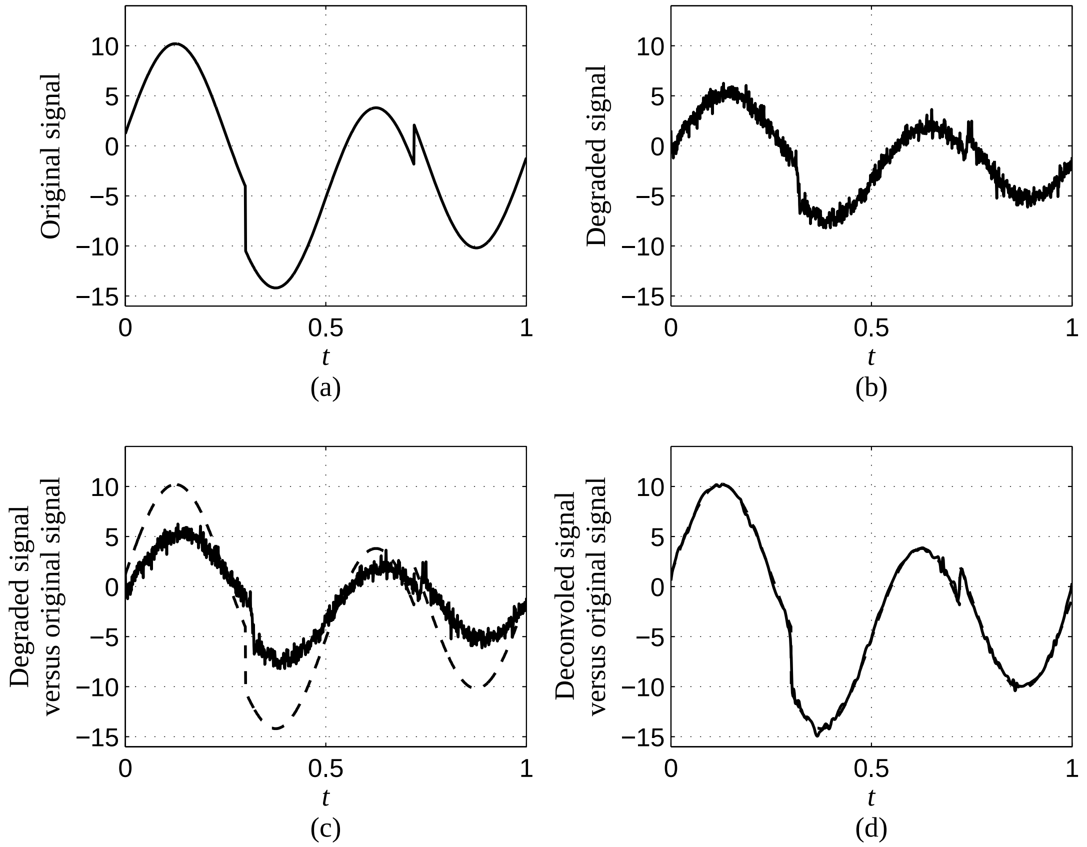


Figure 4.13. (a) Original signal heavisine (b) Distorted and noise-contaminated heavisine (c) Original (dashed) versus degraded (solid) heavisine, SNR = 5.7592 dB. (d) Original (dashed) versus reconstructed (solid) heavisine, SNR = 27.2765 dB.

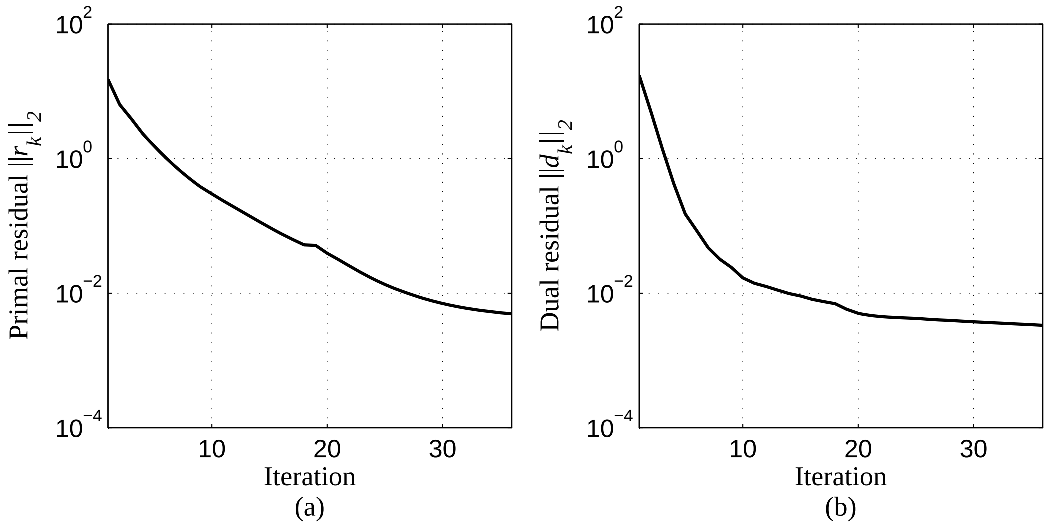


Figure 4.14. (a) Primal residual $\| \boldsymbol{r}_k \|_2$ versus iterations (b) Dual residual $\| \boldsymbol{d}_k \|_2$ versus iterations for Example 4.8.

4.5.2 ADMM for general convex optimization

Consider the constrained convex problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to: } \boldsymbol{x} \in C \end{aligned} \tag{4.94}$$

where $f(\boldsymbol{x})$ is a convex function and C is a convex set representing the feasible region of the problem. The problem in Eq. (4.94) can be formulated as

$$\text{minimize} \quad f(\boldsymbol{x}) + I_C(\boldsymbol{x}) \tag{4.95}$$

where $I_C(\boldsymbol{x})$ is the indicator function associated with set C:

$$I_C(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in C \\ +\infty & \text{otherwise} \end{cases}$$

The problem in Eq. (4.95) can be written as

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) + I_C(\boldsymbol{y}) \\ &\text{subject to: } \boldsymbol{x} - \boldsymbol{y} = \boldsymbol{0} \end{aligned} \tag{4.96}$$

That fits into the ADMM formulation in Eq. (4.76) [8]. The scaled ADMM iterations for Eq. (4.96) are given by

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \tfrac{\alpha}{2} \| \boldsymbol{x} - \boldsymbol{y}_k + \boldsymbol{v}_k \|_2^2 \right] \\ \boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}} \left[I_C(\boldsymbol{y}) + \tfrac{\alpha}{2} \| \boldsymbol{y} - (\boldsymbol{x}_{k+1} + \boldsymbol{v}_k) \|_2^2 \right] \\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1} \end{aligned}$$

where the y-minimization is obtained by minimizing $\| \boldsymbol{y} - (\boldsymbol{x}_{k+1} + \boldsymbol{v}_k) \|_2$ subject to $\boldsymbol{y} \in C$. This means that $\boldsymbol{y}_{k+1}$ can be obtained by projecting $\boldsymbol{x}_{k+1} + \boldsymbol{v}_k$ onto set C. Therefore, the ADMM iterations become

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \tfrac{\alpha}{2} \| \boldsymbol{x} - \boldsymbol{y}_k + \boldsymbol{v}_k \|_2^2 \right] \\ \boldsymbol{y}_{k+1} &= P_C(\boldsymbol{x}_{k+1} + \boldsymbol{v}_k) \\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1} \end{aligned} \tag{4.97a-c}$$

where $P_C(\boldsymbol{z})$ denotes the projection of point $\boldsymbol{z}$ onto convex set C. The projection cab be accomplished by solving the convex problem

$$\begin{aligned} &\text{minimize} \quad \| \boldsymbol{y} - \boldsymbol{z} \|_2 \\ &\text{subject to:} \quad \boldsymbol{y} \in C \end{aligned}$$

Example 4.9
Find a sparse solution of an underdetermined system of linear equation $\boldsymbol{Ax} = \boldsymbol{b}$ by solving the constrained convex problem

$$\begin{aligned} &\text{minimize} \quad \| \boldsymbol{x} \|_1 \\ &\text{subject to:} \quad \boldsymbol{Ax} = \boldsymbol{b} \end{aligned} \tag{4.98}$$

Solution The problem fits into the formulation in Eq. (4.94) with $f(\boldsymbol{x}) = \| \boldsymbol{x} \|_1$ and $C = \{ \boldsymbol{x} : \boldsymbol{Ax} = \boldsymbol{b} \}$. The $\boldsymbol{x}$-minimization step in Eq. (4.97a) becomes

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \left[\| \boldsymbol{x} \|_1 + \tfrac{\alpha}{2} \| \boldsymbol{x} - \boldsymbol{y}_k + \boldsymbol{v}_k \|_2^2 \right]$$

hence

$$\boldsymbol{x}_{k+1} = \mathcal{S}_{1/\alpha}(\boldsymbol{y}_k - \boldsymbol{v}_k) \tag{4.99}$$

where operator $\mathcal{S}$ is defined by (4.68). The $\boldsymbol{y}$-minimization step in Eq. (4.94b) is carried out by solving the simple convex QP problem

$$\begin{aligned} &\text{minimize} \quad \| \boldsymbol{y} - (\boldsymbol{x}_{k+1} + \boldsymbol{v}_k) \|_2 \\ &\text{subject to:} \quad \boldsymbol{Ay} = \boldsymbol{b} \end{aligned} \tag{4.100}$$

whose solution is given by

$$\boldsymbol{y}_{k+1} = \boldsymbol{A}^+ \boldsymbol{b} + \boldsymbol{P}_A(\boldsymbol{x}_{k+1} + \boldsymbol{v}_k) \tag{4.101}$$

where $\boldsymbol{A}^+ = \boldsymbol{A}^T(\boldsymbol{AA}^T)^{-1}$ and $\boldsymbol{P}_A = \boldsymbol{I} - \boldsymbol{A}^T(\boldsymbol{AA}^T)^{-1}\boldsymbol{A}$. To summarize, the ADMM iterations in Eq. (4.97) are realized using Eqs. (4.99), (4.101), and (4.97c).
For illustration purposes, above results were applied to the problem in Eq. (4.98) where $\boldsymbol{A}$ was a

randomly generated matrix of size 20 by 50. A sparse column vector $\boldsymbol{x}_s$ of length 50 was produced by placing six randomly generated nonzero numbers in the zero vector of length 50 for six randomly selected coordinates, see Fig. 4.15a. Vector $\boldsymbol{b}$ was then generated as $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}_s$. In this way, $\boldsymbol{x}_s$ is a known sparse solution of the underdetermined system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. With $\alpha = 0.5$, it took 49 ADMM iterations to converge to a solution $\boldsymbol{x}^*$ which is depicted in Fig. 4.15b where the ADMM-based solution is found to well recover the true sparse solution $\boldsymbol{x}_s$. The l_2 reconstruction error of the solution is found to be $\|\boldsymbol{x}^* - \boldsymbol{x}_s\|_2 = 4.1282 \times 10^{-4}$. ■

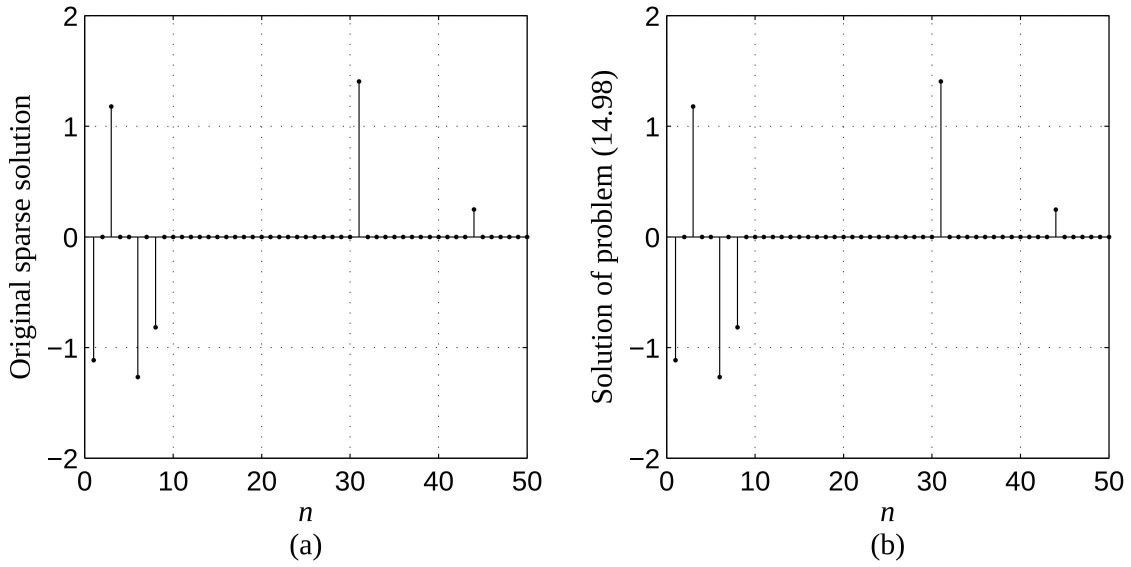


Figure 4.15. (a) Original sparse solution $\boldsymbol{x}_s$ and (b) Solution obtained by solving the problem in Eq. (4.98).

The primal and dual residuals are given by $\boldsymbol{r}_k = \boldsymbol{x}_k - \boldsymbol{y}_k$ and $\boldsymbol{d}_k = -\alpha(\boldsymbol{y}_k - \boldsymbol{y}_{k-1})$, respectively. The profiles of these residuals for the 49 ADMM iterations are shown in Fig. 4.16.

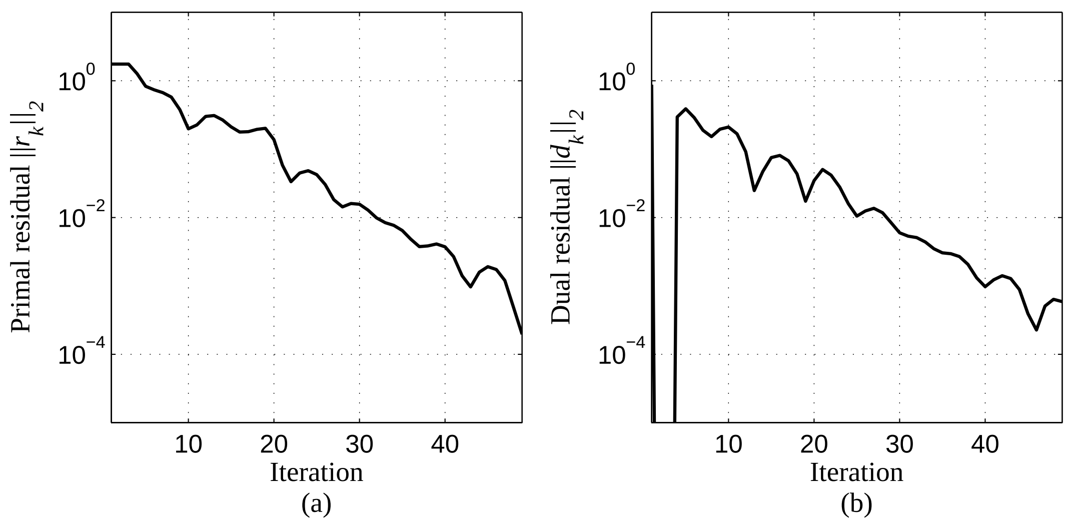


Figure 4.16. (a) Primal and (b) Dual residuals for the problem in Example 4.8.

4.5.3 Alternating minimization algorithm (AMA)

Alternating minimization algorithms (AMA) are developed to solve the problem in Eq. (4.76), namely,

$$\begin{aligned} &\text{minimize } \ f(\boldsymbol{x}) + h(\boldsymbol{y}) \\ &\text{subject to: } \boldsymbol{Ax} + \boldsymbol{By} = \boldsymbol{c} \end{aligned}$$

where one of the objective components, say $f(\boldsymbol{x})$, is strongly convex [10]. Under this assumption, updating $\boldsymbol{x}_k$ in an AMA iteration is simplified relative to an ADMM iteration:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \boldsymbol{\lambda}_k^T \boldsymbol{Ax} \right] \\ \boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}} \left[h(\boldsymbol{y}) + \boldsymbol{\lambda}_k^T \boldsymbol{By} + \tfrac{\alpha}{2} \| \boldsymbol{Ax}_{k+1} + \boldsymbol{By} - \boldsymbol{c} \|_2^2 \right] \qquad (4.102\text{a-c}) \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \alpha(\boldsymbol{Ax}_{k+1} + \boldsymbol{By}_{k+1} - \boldsymbol{c}) \end{aligned}$$

With an argument similar to that used for ADMM iterations, Eqs. (4.102b-c) are shown to imply

$$\nabla h(\boldsymbol{y}_{k+1}) + \boldsymbol{B}^T \boldsymbol{\lambda}_{k+1} = 0$$

Therefore, the KKT conditions in Eq. (4.77c) is satisfied by AMA iterations. Since $\boldsymbol{x}_{k+1}$ minimizes $f(\boldsymbol{x}) + \boldsymbol{\lambda}_k^T \boldsymbol{Ax}$, we have

$$\boldsymbol{0} = \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T \boldsymbol{\lambda}_k = \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T \boldsymbol{\lambda}_{k+1} - \boldsymbol{A}^T(\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k)$$

i.e.

$$\nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T \boldsymbol{\lambda}_{k+1} = \boldsymbol{A}^T(\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k)$$

which, on comparing it with Eq. (4.77b), suggests a *dual residual* as

$$\boldsymbol{d}_k = \boldsymbol{A}^T(\boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k) \qquad (4.103)$$

The *primal residual* for AMA is the same as in ADMM, namely,

$$\boldsymbol{r}_k = \boldsymbol{Ax}_{k+1} + \boldsymbol{By}_{k+1} - \boldsymbol{c} \qquad (4.104)$$

A reasonable stopping criteria for AMA iterations is when both

$$\| \boldsymbol{r}_k \| < \varepsilon_p \ \text{ and } \ \| \boldsymbol{d}_k \| < \varepsilon_d \qquad (4.105)$$

Are satisfied, where ε_p and ε_d are prescribed tolerances for primal and dual residuals, respectively. The method for solving the problem in Eq. (4.76) can now be outlined as an algorithm.

Algorithm 4.13 AMA for the problem in Eq. (4.76)

Step 1 Input parameter $\alpha > 0$, $\boldsymbol{y}_0$, $\boldsymbol{\lambda}_0$, and tolerance $\varepsilon_p > 0$, $\varepsilon_d > 0$.

Set $k = 0$.

Step 2 Compute $\{\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}, \boldsymbol{\lambda}_{k+1}\}$ using Eq. (4.102).

Step 3 Compute $\boldsymbol{d}_k$ and $\boldsymbol{r}_k$ using Eqs. (4.103) and (4.104), respectively.

Step 4 If the conditions in Eq. (4. 105) are satisfied, output ($\boldsymbol{x}_{k+1}$, $\boldsymbol{y}_{k+1}$) as solution and stop; Otherwise, set $k = k + 1$ and repeat from Step 2.

It can be shown [9] that if $f(\boldsymbol{x})$ is strongly convex and $\alpha < m / \rho(\boldsymbol{A}^T\boldsymbol{A})$ where m is the parameter associated with the strong convexity of $f(\boldsymbol{x})$ as seen in Eq. (4.13) and $\rho(\boldsymbol{A}^T\boldsymbol{A})$ denotes the largest eigenvalue of $\boldsymbol{A}^T\boldsymbol{A}$, then the fast AMA algorithm outlined below converges to the solution of Eq. (4.76).

Algorithm 4.14 Accelerated AMA for the problem in Eq. (4.76)

Step 1 Input $\alpha < m / \rho(\boldsymbol{A}^T\boldsymbol{A})$, $\hat{\boldsymbol{\lambda}}_0$, and tolerance $\varepsilon_p > 0$, $\varepsilon_d > 0$.

Set $\boldsymbol{\lambda}_{-1} = \hat{\boldsymbol{\lambda}}_0$, $t_0 = 1$, and $k = 0$.

Step 2 Compute

$$\boldsymbol{x}_k = \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \hat{\boldsymbol{\lambda}}_k^T \boldsymbol{A}\boldsymbol{x} \right]$$

$$\boldsymbol{y}_k = \arg\min_{\boldsymbol{y}} \left[h(\boldsymbol{y}) + \hat{\boldsymbol{\lambda}}_k^T \boldsymbol{B}\boldsymbol{y} + \tfrac{\alpha}{2} \| \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \|_2^2 \right]$$

$$\boldsymbol{\lambda}_k = \hat{\boldsymbol{\lambda}}_k + \alpha(\boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{y}_k - \boldsymbol{c})$$

$$t_{k+1} = \tfrac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$$

$$\hat{\boldsymbol{\lambda}}_{k+1} = \boldsymbol{\lambda}_k + \tfrac{t_k - 1}{t_{k+1}}(\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1})$$

Step 3 If $\| \boldsymbol{A}^T(\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1}) \|_2 < \varepsilon_d$ and $\| \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{y}_k - \boldsymbol{c} \|_2 < \varepsilon_p$, output ($\boldsymbol{x}_k$, $\boldsymbol{y}_k$) as solution and stop; Otherwise, set $k = k + 1$ and repeat from Step 2.

References

[1] R. T. Rockafellar, *Convex Analysis*, Princeton, N.J.: Princeton University Press, 1970.

[2] Y. E. Nesterov, *Introductory Lectures on Convex Optimization – A Basic Course*, Norwell, MA.: Kluwer Academic Publishers, 2004.

[3] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge, UK: Cambridge University Press, 2004.

[4] N. Komodakis and J.-C. Pesquet, "Playing with duality,", *IEEE Signal Processing Magazine*, vol. 21, no. 6, pp. 31-54, Nov. 2015.

[5] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 123–231, 2013.

[6] Y. Nesterov, "Gradient methods for minimizing composite functions," *Math. Programming*, vol. 140, no. 1, pp. 125-161, Aug. 2013.

[7] A. Beck and M. Teboulle, "A fast shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imaging Sciences*, vol. 2, no. 1, pp. 183-202, 2009.

[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[9] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM J. Imaging Sciences*, vol. 7, no. 3, pp. 1588-1623, 2014.

[10] P. Tseng, "Applications of splitting algorithm to decomposition in convex programming and variational inequalities," *SIAM J. Control and Optimization*, vol. 29, no. 1, pp. 119-138, 1991.

Problems

4.1 (a) Find the subdifferential of $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ (Hint: Use the result of Example 4.1).

(b) Find the subdifferential of $f(\boldsymbol{x}) = \|\boldsymbol{x}\|_2$.

(c) Show that the subdifferential of $f(\boldsymbol{x}) = \sum_{i=1}^{m} |\boldsymbol{a}_i^T \boldsymbol{x} - b_i|$ is given by

$$\partial f(\boldsymbol{x}) = \sum_{i \in I_+(\boldsymbol{x})} \boldsymbol{a}_i - \sum_{i \in I_-(\boldsymbol{x})} \boldsymbol{a}_i + \sum_{i \in I_0(\boldsymbol{x})} \text{any } \boldsymbol{g}_i \in [-\boldsymbol{a}_i, \boldsymbol{a}_i] \tag{P4.1}$$

where $I_+(\boldsymbol{x})$, $I_-(\boldsymbol{x})$, and $I_0(\boldsymbol{x})$ are defined by

$$\begin{aligned} I_+(\boldsymbol{x}) &= \{i : \boldsymbol{a}_i^T \boldsymbol{x} - b_i > 0\} \\ I_-(\boldsymbol{x}) &= \{i : \boldsymbol{a}_i^T \boldsymbol{x} - b_i < 0\} \\ I_0(\boldsymbol{x}) &= \{i : \boldsymbol{a}_i^T \boldsymbol{x} - b_i = 0\} \end{aligned}$$

In Eq. (4.1), $\boldsymbol{g}_i \in [-\boldsymbol{a}_i, \boldsymbol{a}_i]$ is understood as $\boldsymbol{g}_i = t\boldsymbol{a}_i$ for some scalar t between –1 and 1 for all i.

(d) Suppose $f(\boldsymbol{x})$ is convex. Show that if $h(\boldsymbol{x}) = f(\boldsymbol{Ax} + \boldsymbol{b})$, then $\partial h(\boldsymbol{x}) = \boldsymbol{A}^T \partial f(\boldsymbol{Ax} + \boldsymbol{b})$.

4.2 Show that subgradient of a convex function is a monotone operator. Namely,

$$(\boldsymbol{g}_x - \boldsymbol{g}_y)^T (\boldsymbol{x} - \boldsymbol{y}) \ge 0$$

where $\boldsymbol{g}_x$ and $\boldsymbol{g}_y$ are any subgradients of $f(\boldsymbol{x})$ at $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively.

4.3 Show that

$$\begin{bmatrix} \partial f(\boldsymbol{x}) \\ -1 \end{bmatrix}^T \left(\begin{bmatrix} \boldsymbol{y} \\ t \end{bmatrix} - \begin{bmatrix} \boldsymbol{x} \\ f(\boldsymbol{x}) \end{bmatrix} \right) \le 0 \quad \forall\, (\boldsymbol{y}, t) \in \text{epi}(f)$$

where function $f(\boldsymbol{x})$ is convex but not necessarily differentiable.

4.4 The *projection* of a point $\boldsymbol{z} \in R^n$ onto a closed convex set C, to be denoted by $P_C(\boldsymbol{z})$, is defined as a point in C, which possesses the smallest Euclidean distance to point $\boldsymbol{z}$ among all points in C. Namely,

$$P_C(\boldsymbol{z}) = \arg\min_{\boldsymbol{x} \in C} \|\boldsymbol{x} - \boldsymbol{z}\|_2$$

(a) Given $C = \{x : a \le x \le b\}$ and $z \in R$, derive a closed-form expression for the projection $P_C(z)$.

(b) Given $C = \left\{ \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} : \frac{(x_1-4)^2}{9} + \frac{(x_2-5)^2}{4} \le 1 \right\}$ and $\boldsymbol{z} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$, compute $P_C(\boldsymbol{z})$.

4.5 Now consider the convex set $C = \{\boldsymbol{x} : \boldsymbol{Ax} = \boldsymbol{b}\}$ where $\boldsymbol{A} \in R^{p \times n}, \boldsymbol{b} \in R^{p \times 1}, p < n$, and $\boldsymbol{A}$ is of full row-rank.

(a) Given a point $\boldsymbol{z} \notin C$, formulate the problem of projecting $\boldsymbol{z}$ onto C as a convex problem.

(b) Deduce a closed-formula for $P_C(\boldsymbol{z})$ by solving the problem in part (a).

(c) Given $C = \{\boldsymbol{x} : \boldsymbol{Ax} = \boldsymbol{b}\}$ where $\boldsymbol{A} = \begin{bmatrix} 1 & 2 & -1 \\ 0 & -1 & 3 \end{bmatrix}$, $\boldsymbol{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ and a point $\boldsymbol{z} = \begin{bmatrix} 1 & -1 & 2 \end{bmatrix}^T$.

Apply the result from part (b) to compute $P_C(\boldsymbol{z})$.

4.6

(a) Given $C = \{\boldsymbol{x} : \boldsymbol{Ax} \le \boldsymbol{b}\}$, formulate the problem of finding $P_C(\boldsymbol{z})$ as a convex minimization problem.

(b) Given $C = \{\boldsymbol{x} : \boldsymbol{Ax} \le \boldsymbol{b}\}$ where

$$\boldsymbol{A} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \\ 2 \\ -2 \\ -3 \\ 6 \end{bmatrix}$$

and a point $\boldsymbol{z} = \begin{bmatrix} 2 & 1 & 4 & 3 \end{bmatrix}^T$. Compute $P_C(\boldsymbol{z})$ by solving the convex problem formulated in part (a).

4.7 Identify which of the following quadratic functions $f_i(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{H}_i \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{p}_i$ are (*a*) convex with Lipschitz continuous gradients and (*b*) strongly convex:

- $\boldsymbol{H}_1 = \begin{bmatrix} 2 & -2 & 0 \\ -2 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{p}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

- $\boldsymbol{H}_2 = \begin{bmatrix} 2 & -2.01 & 0 \\ -2.01 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{p}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

• $\boldsymbol{H}_3 = \begin{bmatrix} 2 & -2 & 0 \\ -2 & 2.01 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \boldsymbol{p}_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

4.8 Recall the *conjugate function* $f^*(\boldsymbol{u})$ of a function $f(\boldsymbol{x})$ is defined as

$$f^*(\boldsymbol{u}) = \sup_{\boldsymbol{x}\in \text{dom}(f)} \left(\boldsymbol{u}^T \boldsymbol{x} - f(\boldsymbol{x}) \right)$$

This exercise problem examines several additional properties of conjugate functions, which are closely related to the notion of Lagrange dual function.

(a) The concept of conjugate function is related to the inverse of the gradient in the following way: one can regard the gradient $\nabla f(\boldsymbol{x})$ as a mapping from a subset of R^n to a subset of R^n and pose a question — given a vector $\boldsymbol{y}$ from R^n, find an $\boldsymbol{x}$ such that $\nabla f(\boldsymbol{x}) = \boldsymbol{u}$. Evidently, this question is about finding inverse mapping of $\nabla f(\boldsymbol{x})$. Show that the solution (if it exists) of this inverse gradient mapping problem is given by

$$\boldsymbol{x} = \arg\max_{\hat{\boldsymbol{x}}} \left(\boldsymbol{u}^T \hat{\boldsymbol{x}} - f(\hat{\boldsymbol{x}}) \right)$$

(b) Show that conjugate function is differentiable and

$$\nabla f^*(\boldsymbol{u}) = \arg\max_{\hat{\boldsymbol{x}}} \left(\boldsymbol{u}^T \hat{\boldsymbol{x}} - f(\hat{\boldsymbol{x}}) \right) \tag{P4.2}$$

Based on the result from part (a) and Eq. (P4.2), show that

$$\nabla f^*(\boldsymbol{u}) = \nabla^{-1} f(\boldsymbol{u})$$

where $\nabla^{-1} f$ denotes the inverse of the graduate mapping of $f(\boldsymbol{x})$.

(c) Consider the problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to: } \boldsymbol{x} = \boldsymbol{0} \end{aligned}$$

Show that the Lagrange dual and conjugate functions of the above problems are related by

$$q(\boldsymbol{\lambda}) = -f^*(-\boldsymbol{\lambda})$$

(d) Show that the Lagrange dual function $q(\boldsymbol{\lambda}, \boldsymbol{\mu})$ and conjugate functions of the convex problem

$$\begin{aligned} &\text{minimize} \qquad f(\boldsymbol{x}) \\ &\text{subject to: } \boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b},\ \boldsymbol{C}\boldsymbol{x} = \boldsymbol{d} \end{aligned}$$

are related by

$$q(\boldsymbol{\lambda},\boldsymbol{\mu}) = -\boldsymbol{b}^T\boldsymbol{\mu} - \boldsymbol{d}^T\boldsymbol{\lambda} - f^*(-\boldsymbol{A}^T\boldsymbol{\mu} - \boldsymbol{C}^T\boldsymbol{\lambda})$$

4.9 Consider the general convex minimization problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to:} \ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ &\qquad\quad c_j(\boldsymbol{x}) \le 0 \ \text{ for } j = 1,2,\ldots,q \end{aligned} \qquad \text{(P4.3a-c)}$$

where functions $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$ are all convex.

(a) Show that the problem in Eq. (P4.3) is equivalent to the problem

$$\begin{aligned} &\text{minimize} \quad F(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{j=1}^{q} I_{\_}(c_j(\boldsymbol{x})) \\ &\text{subject to:} \ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned} \qquad \text{(P4.4a-b)}$$

where $I_{\_}(u)$ denotes the *indicator function* for nonpositive real *u*:

$$I_{\_}(u) = \begin{cases} 0 & \text{if } u \le 0 \\ \infty & \text{if } u > 0 \end{cases} \qquad \text{(P4.5)}$$

(b) Show that the objective function $F(\boldsymbol{x})$ in Eq. (P4.4a) is convex.

4.10 Evaluate and compare function $h(u) = -(1/\tau)\log(-u)$ over $u < 0$ with the indicator function $I_{\_}(u)$ defined in Eq. (P4.5), where $\tau > 0$ is a parameter that controls the accuracy of the approximation:

(a) Show that for a fixed $\tau > 0$ $h(u)$ is convex with respect to u.

(b) Plot the indicator function $I_{\_}(u)$ over $u < 0$.

(c) In the same figure, draw function $h(u)$ over the interval [–2, 0) for $\tau = 0.5$, 1, and 2. What can you conclude from the figures?

(d) Formulate a convex optimization problem whose objective function is differentiable that approximates the problem in Eq. (P4.4).

4.11 Apply Algorithm 4.5 to solve the convex problem

$$\begin{aligned} &\text{minimize} \ f(\boldsymbol{x}) = \frac{\|\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}\|_2^2}{\boldsymbol{c}^T\boldsymbol{x} + d} \\ &\text{subject to:} \ \boldsymbol{c}^T\boldsymbol{x} + d \ge \delta \end{aligned}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 1 & -0.5 \\ 0 & 0.5 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \boldsymbol{c} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}, \quad d = -1$$

and δ is a small positive constant which is set to $\delta = 0.01$.

4.12 (a) Solve the l_1-l_2 problem (which is a special case of the problem in (4.65) with $\boldsymbol{A} = \boldsymbol{I}$)

$$\text{minimize} \quad F(\boldsymbol{x}) = \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{b}\|_2^2 + \mu\|\boldsymbol{x}\|_1$$

where $\boldsymbol{b} \in R^{n\times 1}$ and $\mu > 0$ are known. The solution of the above problem can be expressed in closed form.

(b) Solve the l_1-l_2 problem

$$\underset{\Theta}{\text{minimize}} \quad F(\Theta) = \tfrac{1}{2}\|\Theta - \boldsymbol{B}\|_F^2 + \mu\|\Theta\|_1$$

where $\boldsymbol{B} \in R^{n\times n}$ and $\mu > 0$ are known, $\|\cdot\|_F$ denotes Frobenius norm for matrices, and $\|\Theta\|_1 = \sum_{i=1}^n \sum_{j=1}^n |\theta_{i,j}|$. The solution of the above problem can be expressed in closed form.

4.13 (a) A noise-contaminated image $\boldsymbol{Y}$ of size n by n can be modeled as $\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{W}$ where $\boldsymbol{X}$ is an image to be estimated and $\boldsymbol{W}$ is noise. It is well known that images are sparse in DCT domain: this is to say, although $\boldsymbol{X}$ is not at all sparse, the discrete cosine transform of $\boldsymbol{X}$ becomes sparse. Furthermore, the DCT of image $\boldsymbol{X}$ can be obtained as $\Theta = \boldsymbol{CXC}^T$ where $\boldsymbol{C}$ is a DCT matrix which can be calculated in MATLAB using `C = dctmtx(n)`. Based on these, formulate the denoising problem as hand as the l_1-l_2 minimization problem in Prob. 4.12(b), and derive a closed-form solution based on the results from Prob. 4.12(b).

(b) Now apply the results of part (a) to a particular test image as follows. Download image `boat512.mat` of size 512 by 512 from the class webpage. Denote the image by $\boldsymbol{X}$, i.e., `X = boat512/256`. Construct a noisy image $\boldsymbol{Y}$ by adding a certain amount of noise to image $\boldsymbol{X}$, i.e., $\boldsymbol{Y} = \boldsymbol{X} + \boldsymbol{W}$ where $\boldsymbol{W}$ is constructed as follows:

```
randn('state',29)
W = 0.1*randn(512,512);
Y = X + W;
```

(c) Apply the results from part (a) to denoise image `Y` constructed in part (b). Evaluate the denoising performance in terms of signal-to-noise ratio (SNR) before and after denoising, which are defined by

$$\text{SNR}_{\text{before}} = 20\log_{10}\frac{\|\boldsymbol{X}\|_F}{\|\boldsymbol{Y}-\boldsymbol{X}\|_F} \quad \text{and} \quad \text{SNR}_{\text{after}} = 20\log_{10}\frac{\|\boldsymbol{X}\|_F}{\|\hat{\boldsymbol{X}}-\boldsymbol{X}\|_F}$$

respectively, where $\hat{\boldsymbol{X}}$ denotes the denoised image.

4.14 This exercise problem examines a *signal separation* problem which is to be solved using an *alternating l_1-l_2 minimization* strategy. Suppose the signal measurement that you can access is an additive mixture of two different types of signals with some measurement noise, both signals are useful and you want to separate the two signals from each other. The signal model in this case is given by

$$\boldsymbol{u} = \boldsymbol{x}_1 + \boldsymbol{x}_2 + \boldsymbol{w} \tag{P4.6}$$

An effective technique to solve this problem is to use an alternating l_1-l_2 minimization technique as described below. First, for each $\boldsymbol{x}_i$, one chooses a coordinate transformation $\boldsymbol{T}_i$ so that $\boldsymbol{x}_i$ becomes sparse under $\boldsymbol{T}_i$. That is, one finds orthogonal $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ such that $\boldsymbol{\theta}_1 = \boldsymbol{T}_1\boldsymbol{x}_1$ and $\boldsymbol{\theta}_2 = \boldsymbol{T}_2\boldsymbol{x}_2$ with $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ sparse. Under these circumstances, the signal model in Eq. (P4.6) becomes

$$\boldsymbol{u} = \boldsymbol{T}_1^T\boldsymbol{\theta}_1 + \boldsymbol{T}_2^T\boldsymbol{\theta}_2 + \boldsymbol{w} \tag{P4.7}$$

A natural way to recover $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ in Eq. (P4.7) is to solve the constrained problem

$$\begin{aligned}&\text{minimize} \quad \|\boldsymbol{\theta}_1\|_1 + \|\boldsymbol{\theta}_2\|_1\\ &\text{subject to: } \|\boldsymbol{T}_1^T\boldsymbol{\theta}_1 + \boldsymbol{T}_2^T\boldsymbol{\theta}_2 - \boldsymbol{u}\|_2 \le \varepsilon\end{aligned}$$

which can be converted into the unconstrained problem

$$\text{minimize} \quad \tfrac{1}{2}\|\boldsymbol{T}_1^T\boldsymbol{\theta}_1 + \boldsymbol{T}_2^T\boldsymbol{\theta}_2 - \boldsymbol{u}\|_2^2 + \mu\|\boldsymbol{\theta}_1\|_1 + \mu\|\boldsymbol{\theta}_2\|_1 \tag{P4.8}$$

with an adequate regularization parameter μ > 0. The second step is to carry out a procedure where the l_1-l_2 minimization problem in Eq. (P4.8) is solved by alternating shrinkage.

(a) Show that for a fixed $\boldsymbol{\theta}_2$, the global minimizer of problem in Eq. (P4.8) is given by

$$\boldsymbol{\theta}_1 = \mathcal{S}_\mu\left(\boldsymbol{T}_1(\boldsymbol{u} - \boldsymbol{T}_2^T\boldsymbol{\theta}_2)\right) \tag{P4.9}$$

(b) Show that for a fixed $\boldsymbol{\theta}_1$, the global minimizer of the problem in Eq. (P4.8) is given by

$$\boldsymbol{\theta}_2 = \mathcal{S}_\mu\left(\boldsymbol{T}_2(\boldsymbol{u} - \boldsymbol{T}_1^T\boldsymbol{\theta}_1)\right) \tag{P4.10}$$

Based on Eqs. (P4.9) and (P4.10), develop an iterative formula that updates the *k*th iterate $\{\boldsymbol{\theta}_1^{(k)}, \boldsymbol{\theta}_2^{(k)}\}$ to $\{\boldsymbol{\theta}_1^{(k+1)}, \boldsymbol{\theta}_2^{(k+1)}\}$.

(c) Based on the formula developed in part (b), solve the signal separation problem where the data in model (P4.7) are generated in MATLAB as follows.

(i) Signals $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are generated as

```
t = 0:1/511:1;
x1 = -1.8*sin(2*pi*1.2*t)+1.2*cos(2*pi*1.8*t)-sin(2*pi*3.1*t)-0.8;
x1 = x1(:);
x2 = zeros(512,1);
rand('state',14);
r = randperm(512);
randn('state',28);
x2(r(1:13)) = 2*randn(13,1);
```

(ii) Matrix $\boldsymbol{T}_1$ is an DCT matrix and matrix $\boldsymbol{T}_2$ is an identity matrix which are generated as

```
T1 = dctmtx(512);
T2 = eye(512);
```

(iii) Noise $\boldsymbol{w}$ is generated as

```
randn('state',10)
w = 0.01*randn(512,1);
```

(iv) Using the above data, you can easily generate the measurement vector $\boldsymbol{u}$ using Eq. (P4.6).

Now start with $\{\boldsymbol{x}_1^{(0)}, \boldsymbol{x}_2^{(0)}\} = \{\mathbf{0}, \mathbf{0}\}$, perform a total of 31 iterations based on part (c) to obtain estimates $\tilde{\boldsymbol{x}}_1 \triangleq \boldsymbol{x}_1^{(31)}$ and $\tilde{\boldsymbol{x}}_2 \triangleq \boldsymbol{x}_2^{(31)}$. It is recommended that μ = 0.01 is used in your iterations.

To evaluate separation performance, you are required to do the following:

- Generate a plot that displays the data curve $\boldsymbol{u}$ versus t;
- Generate a plot to show both $\boldsymbol{x}_1$ and its estimate $\tilde{\boldsymbol{x}}_1$;

- Generate a plot to show both $\boldsymbol{x}_2$ and its estimate $\tilde{\boldsymbol{x}}_2$;
- Compute average 2-norm estimation errors $\|\tilde{\boldsymbol{x}}_1 - \boldsymbol{x}_1\|_2 / \sqrt{n}$ and $\|\tilde{\boldsymbol{x}}_2 - \boldsymbol{x}_2\|_2 / \sqrt{n}$ with n = 512;
- Comment on the results you obtained.

4.15 Find a point in the intersection of two closed nonempty convex sets $\mathcal{S}_1$ and $\mathcal{S}_2$. Let $I_{\mathcal{S}1}$ and $I_{\mathcal{S}2}$ be the indicator functions of $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. An ADMM formulation of the problem is given by [8]

$$\begin{aligned}&\text{minimize } I_{\mathcal{S}_1}(\boldsymbol{x}) + I_{\mathcal{S}_2}(\boldsymbol{y})\\ &\text{subject to: } \boldsymbol{x} - \boldsymbol{y} = \boldsymbol{0}\end{aligned} \qquad \text{(P4.11)}$$

(a) Write down the scaled form of ADMM iterations for the problem in Eq. (P4.11). (Hint: the *x*- and *y*-minimization steps can be expressed explicitly in term of projections onto sets $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively.)

(b) Apply the ADMM iterations obtained from part (a) to obtain a point in the intersection of sets $\mathcal{S}_1$ and $\mathcal{S}_2$ that are characterized below.

$$\mathcal{S}_1 : \begin{cases} -x_1 \le 0 \\ -x_2 \le 0 \\ \frac{1}{2}x_1 + x_2 - 1 \le 0 \end{cases} \qquad \mathcal{S}_2 : \begin{cases} 2x_1 - x_2 - 1.6 \le 0 \\ -2x_1 - x_2 + 2.4 \le 0 \\ x_2 - 3 \le 0 \end{cases}$$

4.16 Consider the problem of finding a point that lies in the intersection of K closed convex sets $\{\mathcal{S}_i, i = 1, 2, \ldots, K\}$ with $K > 2$.

(a) Extend the ADMM-based method developed in Prob. 4.13 to the problem of finding a point that lies in the intersection of K closed convex sets $\{\mathcal{S}_i, i = 1, 2, \ldots, K\}$ with $K > 2$.

(b) Apply the method developed in part (a) to find a feasible point for linear inequality constraints $\boldsymbol{Ax} \le \boldsymbol{b}$ where $\boldsymbol{A}$ and $\boldsymbol{b}$ are given in Prob. 4.6(b).

(c) Apply the method in part (b) to find a *strictly* feasible point for linear inequality constraints $\boldsymbol{Ax} \le \boldsymbol{b}$ where $\boldsymbol{A}$ and $\boldsymbol{b}$ are given in Prob. 4.6(b).

4.17 Consider the convex problem

$$\text{minimize } f(\boldsymbol{x}) = \sum_{i=1}^{K} f_i(\boldsymbol{x}) \qquad \text{(P4.12)}$$

where each $f_i(\boldsymbol{x})$ is convex and assumes value $+\infty$ for $\boldsymbol{x}$ that violates a constraint. The problem in Eq. (P4.12) arises in several applications, especially in the area of modeling, data analysis, and data processing, where often times the number of terms in the objective function, *K*, is large. The problem at hand can be re-formulated as

$$\begin{aligned}&\text{minimize } f(\boldsymbol{x}) = \sum_{i=1}^{K} f_i(\boldsymbol{x}_i)\\ &\text{subject to: } \boldsymbol{x}_i - \boldsymbol{y} = 0 \quad \text{for } i = 1, 2, \ldots, K\end{aligned} \qquad \text{(P4.13)}$$

which is referred to as the *global consensus* problem in that all local variables $\boldsymbol{x}_i$ are constrained to be equal [8]. This re-formulation is of interest as it allows to solve the problem in Eq. (P4.12) in a way that each term in $f(\boldsymbol{x})$ is dealt with by a local processing unit when ADMM is applied to the problem in Eq. (P4.13). Write explicitly the ADMM iterations for the problem in Eq. (P4.13).

Chapter 5 Algorithms for General Nonconvex Problems

5.1 Introduction

There are many optimization problems of practical importance, where the objective function or constraints or both are nonconvex. For these problems, the algorithms developed in early chapters for convex problems are obviously not directly applicable. In this chapter we refer these problems as general optimization problems. One methodology that turns out to be particularly effective in dealing with a general optimization problem is to tackle it in a sequential manner, where one builds a convex model, often called a convex subproblem, that well approximates the original problem in a local region surrounding a known iterate in the space of the design variables. The subproblem is then solved using an appropriate convex algorithm and the solution obtained serves as a new iterate to initiate the next convex subproblem. In this way, one creates a sequence of convex models, each is valid only in a vicinity of a particular iterate, yielding a sequence of iterates that will hopefully converge to a solution of the original problem. Since the original problem is not convex, an algorithm based on such sequential convex approximation is heuristic in nature and is not guaranteed to work. On the other hand, however, in practice heuristic methodologies of this type often work well for many general optimization problems, especially when an appropriate initial point can be identified. In Secs. 5.2 and 5.3, we explore two sequential methods for the general optimization problems, known as sequential convex programming (SCP) and sequential quadratic programming (SQP), respectively. In Secs. 5.4, we study a specific SCP method that is based on a convexification technique known as convex-concave procedure. In Sec. 5.5, the ADMM studied in Chapter 4 is extended to the general optimization problems.

5.2 Sequential Convex Programming

The *sequential convex programming* (SCP) [1] is one of the optimization techniques that implement the methodology outlined in the introduction. Throughout we assume that the objective function and those involved in the constraints are twice continuously differentiable.

5.2.1 The Principle of SCP

Consider the constrained problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to:} \ a_i(\boldsymbol{x}) = 0 \ \text{ for } \ i = 1, 2, \ldots, p \\ &\qquad\qquad\quad c_j(\boldsymbol{x}) \le 0 \ \text{ for } \ j = 1, 2, \ldots, q \end{aligned} \tag{5.1a-c}$$

where $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$ are twice continuously differentiable but may not be convex, and $a_i(\boldsymbol{x})$ are continuously differentiable but may not be affine. Suppose we are in the *k*th iteration of an SCP algorithm, where iterate $\boldsymbol{x}_k$ is known and is to be updated to the next iterate $\boldsymbol{x}_{k+1}$. We begin by defining a region surrounding $\boldsymbol{x}_k$, in which a convex subproblem is solved to produce point $\boldsymbol{x}_{k+1}$. In the literature this local region in called a *trust region* and is usually characterized either as a bounding box defined by

$$\mathcal{R}_\rho^{(k)} = \{\boldsymbol{x} : |x_i - x_i^{(k)}| \le \rho_i \text{ for } i = 1, 2, \ldots, n\} \tag{5.2}$$

where $x_i^{(k)}$ denotes the *i*th component of $\boldsymbol{x}_k$, or as a "ball" in the *n*-dimensional Euclidean space defined by

$$\mathcal{R}_\rho^{(k)} = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}_k\| \le \rho\} \tag{5.3}$$

In the trust region functions $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$ are approximated by convex functions $\tilde{f}(\boldsymbol{x})$ and $\tilde{c}_j(\boldsymbol{x})$, respectively, and functions $a_i(\boldsymbol{x})$ are approximated by affine functions $\tilde{a}_i(\boldsymbol{x})$. It is reasonable to expect that within $\mathcal{R}_\rho^{(k)}$ the *convex* subproblem

$$\begin{array}{ll} \text{minimize} & \tilde{f}(\boldsymbol{x}) \\ \text{subject to:} & \tilde{a}_i(\boldsymbol{x}) = 0 \text{ for } i = 1, 2, \ldots, p \\ & \tilde{c}_j(\boldsymbol{x}) \le 0 \text{ for } j = 1, 2, \ldots, q \\ & \boldsymbol{x} \in \mathcal{R}_\rho^{(k)} \end{array} \tag{5.4a-d}$$

approximates the original problem in Eq. (5.1). Hence it is reasonable to take the solution of the convex subproblem in Eq. (5.4) as the next iterate $\boldsymbol{x}_{k+1}$. Having generated $\boldsymbol{x}_{k+1}$, a new trust region $\mathcal{R}_\rho^{(k+1)}$ centered at $\boldsymbol{x}_{k+1}$ can be defined by Eq. (5.2) or (5.3), and convex approximation of functions $f(\boldsymbol{x})$, $c_j(\boldsymbol{x})$, and $a_i(\boldsymbol{x})$ in $\mathcal{R}_\rho^{(k+1)}$ can be performed. Based on this, a convex subproblem similar to that in Eq. (5.4) can be formulated and its solution is taken to be the new iterate $\boldsymbol{x}_{k+2}$. This procedure continues until a certain convergence criterion is met.

5.2.2 Convex approximation of $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$ and affine approximation of $a_i(\boldsymbol{x})$

Convex approximation based on Taylor expansion

To describe convex approximation of a differentiable function, we need a concept called *nonnegative part* of a symmetric matrix. Let $\boldsymbol{M}$ be an $n \times n$ symmetric matrix and $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ be its eigen-decomposition where matrix $\boldsymbol{U}$ is orthogonal whose columns are the eigenvectors of $\boldsymbol{M}$ and matrix Λ is diagonal whose diagonal elements are the eigenvalues of $\boldsymbol{M}$ that are arranged in descending order, namely $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ with $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n$. If all eigenvalues of $\boldsymbol{M}$ are nonnegative, then the nonnegative part of $\boldsymbol{M}$, denoted by $(\boldsymbol{M})_+$, is $\boldsymbol{M}$ itself, i.e. $(\boldsymbol{M})_+ = \boldsymbol{M}$. If $\boldsymbol{M}$ has *r* negative eigenvalues, i.e.,

$0 > \lambda_{n-r+1} \geq \cdots \geq \lambda_n$, then $(\boldsymbol{M})_+ = \boldsymbol{U}\tilde{\Lambda}\boldsymbol{U}^T$ where $\tilde{\Lambda} = \text{diag}\{\lambda_1, \ldots, \lambda_{n-r}, 0, \ldots, 0\}$.

Using the above notion, convex approximation of $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$, denoted by $\tilde{f}(\boldsymbol{x})$ and $\tilde{c}_j(\boldsymbol{x})$ respectively, can be obtained by Taylor expansions of the respective functions at point $\boldsymbol{x}_k$ and, if necessary, modifying the Hessian by taking its nonnegative part. This yields

$$\begin{aligned} \tilde{f}(\boldsymbol{x}) &= f(\boldsymbol{x}_k) + \nabla^T f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) + \tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T \boldsymbol{H}_k (\boldsymbol{x} - \boldsymbol{x}_k) \\ \tilde{c}_j(\boldsymbol{x}) &= c_j(\boldsymbol{x}_k) + \nabla^T c_j(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) + \tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T \boldsymbol{P}_{j,k} (\boldsymbol{x} - \boldsymbol{x}_k) \end{aligned} \tag{5.5a-b}$$

where $\boldsymbol{H}_k = \left(\nabla^2 f(\boldsymbol{x}_k)\right)_+$ and $\boldsymbol{P}_{j,k} = \left(\nabla^2 c_j(\boldsymbol{x}_k)\right)_+$ are nonnegative parts of Hessian $\nabla^2 f(\boldsymbol{x}_k)$ and $\nabla^2 c_j(\boldsymbol{x}_k)$, respectively.

Affine approximation of function $a_i(\boldsymbol{x})$, denoted by $\tilde{a}_i(\boldsymbol{x})$, can be obtained by the 1st-order Taylor expansion of $a_i(\boldsymbol{x})$ at $\boldsymbol{x}_k$ as

$$\tilde{a}_i(\boldsymbol{x}) = a_i(\boldsymbol{x}_k) + \nabla^T a_i(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) \tag{5.5c}$$

It is intuitively clear that the size of the feasible region $\mathcal{R}_\rho^{(k)}$, which is determined by $\{\rho_i, i = 1, 2, \ldots, n\}$ in the case of Eq. (5.2) or by a single scalar ρ in the case of Eq. (5.3), affects the accuracy of approximation made by Eq. (5.5) hence the performance of the SCP algorithm: if $\mathcal{R}_\rho^{(k)}$ is too large, the convex models in Eq. (5.5) are less accurate hence the solution of the convex subproblem will perform poorly; if $\mathcal{R}_\rho^{(k)}$ is too small, the convex models become very accurate, but more iterations are required for the SCP algorithm to converge.

Particle method

There are also alternative convex approximations of $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$, and affine approximations of $a_i(\boldsymbol{x})$. One of them is the so-called *particle method*. In a particle method, a function $f(\boldsymbol{x})$ is approximated by a convex (or affine) function by the following steps:

(1) Choose a set of points $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_K$ from region $\mathcal{R}_\rho^{(k)}$.

(2) Evaluate $y_i = f(\boldsymbol{z}_i)$ for $i = 1, 2, \ldots, K$.

(3) Fit data $\{(\boldsymbol{z}_i, y_i), i = 1, 2, \ldots, K\}$ with a convex (or affine) function by convex optimization.

The last step can for example be implemented by solving the SDP problem

$$\begin{aligned}&\underset{\boldsymbol{H},\boldsymbol{p},r}{\text{minimize}} \quad \sum_{i=1}^{K}\left[(\boldsymbol{z}_i-\boldsymbol{x}_k)^T\boldsymbol{H}(\boldsymbol{z}_i-\boldsymbol{x}_k)+\boldsymbol{p}^T(\boldsymbol{z}_i-\boldsymbol{x}_k)+r-y_i\right]^2\\&\text{subject to: } \boldsymbol{H}\succeq\boldsymbol{0}\end{aligned}$$

where the variables are $\boldsymbol{H}$, $\boldsymbol{p}$, and r. The convex approximation of $f(\boldsymbol{x})$ in this case is taken to be

$$\tilde{f}(\boldsymbol{x})=(\boldsymbol{x}-\boldsymbol{x}_k)^T\boldsymbol{H}(\boldsymbol{x}-\boldsymbol{x}_k)+\boldsymbol{p}^T(\boldsymbol{x}-\boldsymbol{x}_k)+r \tag{5.6a}$$

This convex approximation obviously depends on the current iterate $\boldsymbol{x}_k$ as well as the trust region $\mathcal{R}_\rho^{(k)}$. Note that unlike the Taylor-expansion based method, the particle method can handle nonsmooth functions as well as smooth functions whose derivatives are difficult to compute.

For function $a_i(\boldsymbol{x})$, its affine approximation of the form

$$\tilde{a}_i(\boldsymbol{x})=\boldsymbol{p}^T(\boldsymbol{x}-\boldsymbol{x}_k)+r \tag{5.6b}$$

can be obtained by solving the least squares problem

$$\underset{\boldsymbol{p},r}{\text{minimize}} \quad \sum_{i=1}^{K}\left[(\boldsymbol{p}^T(\boldsymbol{z}_i-\boldsymbol{x}_k)+r-y_i\right]^2$$

Other options

There are techniques other than those specified in Eqs. (5.5) and (5.6) to approximate nonconvex functions by convex ones. If a nonconvex function $f(\boldsymbol{x})$ assumes the form $f(\boldsymbol{x})=f_c(\boldsymbol{x})+f_{nc}(\boldsymbol{x})$ where $f_c(\boldsymbol{x})$ is convex while $f_{nc}(\boldsymbol{x})$ is not convex, then a natural convex approximation of $f(\boldsymbol{x})$ is

$$\tilde{f}(\boldsymbol{x})=f_c(\boldsymbol{x})+\tilde{f}_{nc}(\boldsymbol{x}) \tag{5.7a}$$

where $\tilde{f}_{nc}(\boldsymbol{x})$ is a convex quadratic approximation of $f_{nc}(\boldsymbol{x})$ that can be obtained based on Taylor expansion of $f_{nc}(\boldsymbol{x})$. In doing so, we obtain

$$\tilde{f}_{nc}(\boldsymbol{x})=f_{nc}(\boldsymbol{x}_k)+\nabla^T f_{nc}(\boldsymbol{x}_k)(\boldsymbol{x}-\boldsymbol{x}_k)+\tfrac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_k)^T\boldsymbol{H}_k(\boldsymbol{x}-\boldsymbol{x}_k) \tag{5.7b}$$

where $\boldsymbol{H}_k=\left(\nabla^2 f_{nc}(\boldsymbol{x}_k)\right)_+$. In the case where the nonnegative part of $f_{nc}(\boldsymbol{x})$ is zero, $\tilde{f}_{nc}(\boldsymbol{x})$ is reduced to a linear approximation given by

$$\tilde{f}_{nc}(\boldsymbol{x})=f_{nc}(\boldsymbol{x}_k)+\nabla^T f_{nc}(\boldsymbol{x}_k)(\boldsymbol{x}-\boldsymbol{x}_k) \tag{5.8}$$

Approximating a nonconvex function by Taylor-expansion based linear function in Eq. (5.8) will be a starting point for a method called convex-concave procedure which we shall study in Sec. 5.4. We remark that because of the presence of $f_c(\boldsymbol{x})$, the convex approximation $\tilde{f}(\boldsymbol{x})$ in Eq. (5.7a) is not necessarily quadratic. An SCP algorithm is now outlined as follows.

Algorithm 5.1 Sequential convex programming for problem (5.1)

Step 1 Input feasible initial point $\boldsymbol{x}_0$ for Eq. (5.1), $\{\rho_i, i = 1,2,...,n\}$, and a tolerance ε > 0. Set k = 0.

Step 2 Using Eq. (5.5) to construct functions $\tilde{f}(\boldsymbol{x})$, $\tilde{a}_i(\boldsymbol{x})$ for $1 \le i \le p$, and $\tilde{c}_j(\boldsymbol{x})$ for $1 \le j \le q$.

Step 3 Use $\boldsymbol{x}_k$ as initial point to solve the convex problem in Eq. (5.4) where $\mathcal{R}_\rho^{(k)}$ is defined by Eq. (5.2). Denote the solution obtained by $\boldsymbol{x}_{k+1}$.

Step 4 If $\| \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \| < \varepsilon$, stop and output $\boldsymbol{x}_{k+1}$ as solution; otherwise set $k = k + 1$ and repeat from Step 2.

Variants of the algorithm can be made by using different options of convex approximation of the functions involved and the way the trust region is constructed.

Example 5.1 Apply Algorithm 5.1 to solve the constrained problem

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) = \ln(1 + x_1^2) + x_2 \\ \text{subject to:} \quad & a(\boldsymbol{x}) = (1 + x_1^2)^2 + x_2^2 - 4 = 0 \\ & c(\boldsymbol{x}) = x_1 + x_2^2 + 0.3 \le 0 \end{aligned}$$

Solution

At $\boldsymbol{x}_k$, from Eq. (5.5a) we have

$$\tilde{f}(\boldsymbol{x}) = f(\boldsymbol{x}_k) + \nabla^T f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) + \tfrac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T \boldsymbol{H}_k (\boldsymbol{x} - \boldsymbol{x}_k)$$

where

$$\nabla f(\boldsymbol{x}_k) = \begin{bmatrix} \frac{2x_1^{(k)}}{1+\left(x_1^{(k)}\right)^2} \\ 1 \end{bmatrix}, \quad \boldsymbol{H}_k = \begin{bmatrix} \frac{2\left(1-\left(x_1^{(k)}\right)^2\right)_+}{\left(1+\left(x_1^{(k)}\right)^2\right)^2} & 0 \\ 0 & 0 \end{bmatrix}$$

and $\tilde{c}(\boldsymbol{x}) = c(\boldsymbol{x})$ because $c(\boldsymbol{x})$ is quadratic and convex. The affine approximation of $a(\boldsymbol{x})$ is given by Eq. (5.5b), namely

$$\tilde{a}(\boldsymbol{x}) = a(\boldsymbol{x}_k) + \nabla^T a(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k)$$

where $\nabla a(\boldsymbol{x}_k) = \left[4x_1^{(k)}\left(1 + (x_1^{(k)})^2\right) \quad 2x_2^{(k)}\right]^T$. With $\boldsymbol{x}_0$ = [–1.5 1]^T, $\rho_i = 0.4$ for i = 1, 2, and $\varepsilon = 10^{-6}$, it took the algorithm 8 iterations to converge a solution $\boldsymbol{x}^*$ = [–0.91624199 –0.78501082]^T. Fig. 5.1 shows a trajectory of the first eight iterates and a profile of the objective function versus iteration. The objective function at the solution point was found to be

$$f^* = -\,0.17551736.$$

Satisfaction of constraints were measured by the value of $\boldsymbol{a}(\boldsymbol{x}^*)$ which was found to be less than 10^{-15}, and $c(\boldsymbol{x}^*) = -2.0603\times10^{-9}$ which means that $c(\boldsymbol{x}^*) \leq 0$ is satisfied and $c(\boldsymbol{x})$ is practically active at $\boldsymbol{x}^*$. ■

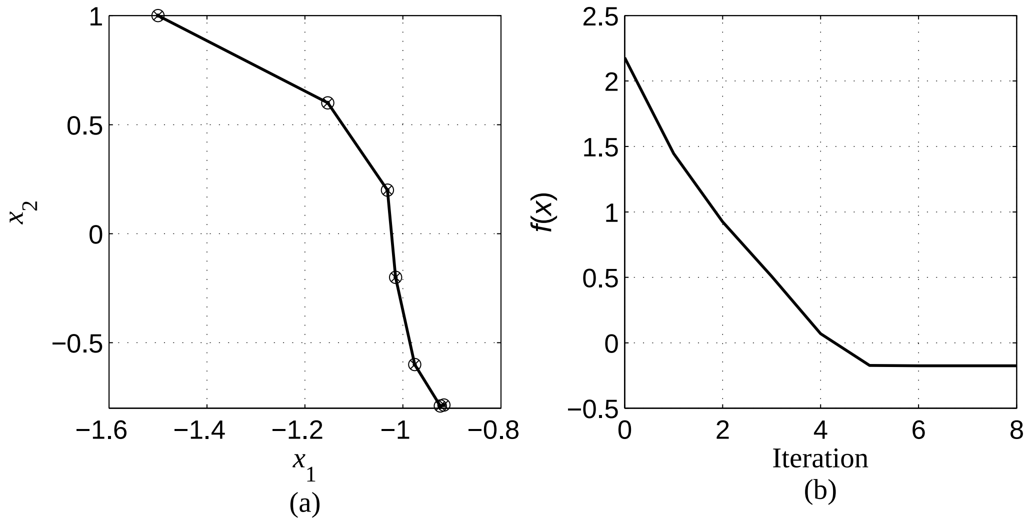


Figure 5.1. (a) First eight iterations produced by Algorithm 5.1 (b) objective function versus iteration for Example 5.1.

5.2.3 An exact penalty formulation

A weak point of Algorithm 5.1 is that the convex formulation in Eq. (5.4) is not guaranteed to be feasible. The exact penalty formulation described below eliminates this difficulty [1]. The method consists of two steps. First, the problem in Eq. (5.1) is reformulated as the unconstrained problem

$$\text{minimize} \quad F_\tau(\boldsymbol{x}) = f(\boldsymbol{x}) + \tau\left(\sum_{i=1}^{p} |a_i(\boldsymbol{x})| + \sum_{j=1}^{q} c_j(\boldsymbol{x})_+\right) \tag{5.9}$$

where $c_j(\boldsymbol{x})_+ = \max\{c_j(\boldsymbol{x}), 0\}$ and τ > 0 weighs the second term which represents the amount of penalty for violation of the constraints. The use of a larger τ simply means a more serve penalty. It can readily be shown that the solution of the problem in Eq. (5.9) also solves the original problem in Eq. (5.1) if τ is sufficiently large. Second, the problem in Eq. (5.9) is solved in a sequential manner where each subproblem is given by

$$\begin{aligned}&\text{minimize} \quad \tilde{F}_\tau(\boldsymbol{x}) = \tilde{f}(\boldsymbol{x}) + \tau\left(\sum\nolimits_{i=1}^{p} |\tilde{a}_i(\boldsymbol{x})| + \sum\nolimits_{j=1}^{q} \tilde{c}_j(\boldsymbol{x})_+\right)\\ &\text{subject to:} \quad \boldsymbol{x} \in \mathcal{R}_\rho^{(k)}\end{aligned} \qquad (5.10\text{a-b})$$

with a fixed value τ and its solution is used to initiate the process of solving the next subproblem in Eq. (5.10) with an increased value τ. Note that the formulation in Eq. (5.10) is always convex and has no feasibility issue because our choice of the initial point ensures that it is always inside the current trust region $\mathcal{R}_\rho^{(k+1)}$.

As mentioned earlier, it is of importance that the trust region used in each subproblem retains an appropriate size. The algorithm below describes a technique to update the size of the trust region in each subproblem.

Algorithm 5.2 Updating size of trust-region in Eq. (5.2)

Step 1 Input iterate $\boldsymbol{x}_k$ and current $\boldsymbol{\rho}_k$. Set constants $\alpha = 0.1, \beta_s = 1.1,$ and $\beta_f = 0.5.$

Step 2 Let $\tilde{\boldsymbol{x}}$ be the solution of the problem in Eq. (5.10), evaluate predicted decrease

$$\tilde{\delta} = F_\tau(\boldsymbol{x}_k) - \tilde{F}_\tau(\tilde{\boldsymbol{x}})$$

and actual decrease

$$\delta = F_\tau(\boldsymbol{x}_k) - F_\tau(\tilde{\boldsymbol{x}})$$

Step 3 If $\delta \ge \alpha\tilde{\delta}$, indicating the actual decrease is greater than a fraction α of the predicted decrease, then set $\boldsymbol{x}_{k+1} = \tilde{\boldsymbol{x}}$ and $\boldsymbol{\rho}_{k+1} = \beta_s \boldsymbol{\rho}_k$ to increase the trust region.

Step 4 If $\delta < \alpha\tilde{\delta}$, indicating the actual decrease is less than a fraction α of the predicted decrease, then set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\boldsymbol{\rho}_{k+1} = \beta_f \boldsymbol{\rho}_k$ to decrease the trust region.

An algorithm based on the exact penalty formulation in Eq. (5.10) is outlined below.

Algorithm 5.3 Sequential convex programming for problem (5.1) based on exact penalty

Step 1 Input feasible initial point $\boldsymbol{x}_0$, $\boldsymbol{\rho}_0 = [\rho_1 \quad \rho_2 \quad \cdots \quad \rho_n]^T$, and number of iterations *K*. Set constants $\alpha = 0.1, \beta_s = 1.1,$ and $\beta_f = 0.5.$ Set $k = 0$.

Step 2 Use Eq. (5.5) to construct functions $\tilde{f}(\boldsymbol{x})$, $\tilde{a}_i(\boldsymbol{x})$ for $1 \le i \le p$, and $\tilde{c}_j(\boldsymbol{x})$ for $1 \le j \le q$.

Step 3 Use $\boldsymbol{x}_k$ as initial point to solve the convex subproblem in Eq. (5.10) where $\mathcal{R}_\rho^{(k)}$ is defined by Eq. (5.2). Denote the solution obtained by $\tilde{\boldsymbol{x}}$.

Step 4 Apply Algorithm 5.2 to obtain $\boldsymbol{x}_{k+1}$ and $\boldsymbol{\rho}_{k+1}$.

Step 5 If $k + 1 = K$, stop and output $\boldsymbol{x}_{k+1}$ as solution; otherwise set $k = k + 1$ and repeat from Step 2.

Example 5.2 Apply Algorithm 5.3 to solve the constrained problem in Example 5.1, i.e.,

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) = \ln(1+x_1^2) + x_2 \\ &\text{subject to:} && a(\boldsymbol{x}) = (1+x_1^2)^2 + x_2^2 - 4 = 0 \\ & && c(\boldsymbol{x}) = x_1 + x_2^2 + 0.3 \le 0 \end{aligned}$$

Solution

With $\boldsymbol{\rho}_0 = [0.1 \quad 0.1]^T$, $\tau = 20$, and the same initial point $\boldsymbol{x}_0 = [-1.5 \ 1]^T$ as in Example 5.1, it took Algorithm 5.3 12 iterations to converge to a solution $\boldsymbol{x}^* = [-0.91628100 \quad -0.78503566]^T$. Fig. 5.2 shows a trajectory of the first 12 iterates and a profile of the objective function versus iteration. The objective function at the solution point was found to be

$$f^* = -0.17550334.$$

Satisfaction of constraints were measured by the value of $\boldsymbol{a}(\boldsymbol{x}^*)$ which was found to be 3.0201×10^{-4} and $c(\boldsymbol{x}^*) = -9.9360 \times 10^{-9}$ which means that $c(\boldsymbol{x}^*) \le 0$ is satisfied and $c(\boldsymbol{x})$ is practically active at $\boldsymbol{x}^*$. ■

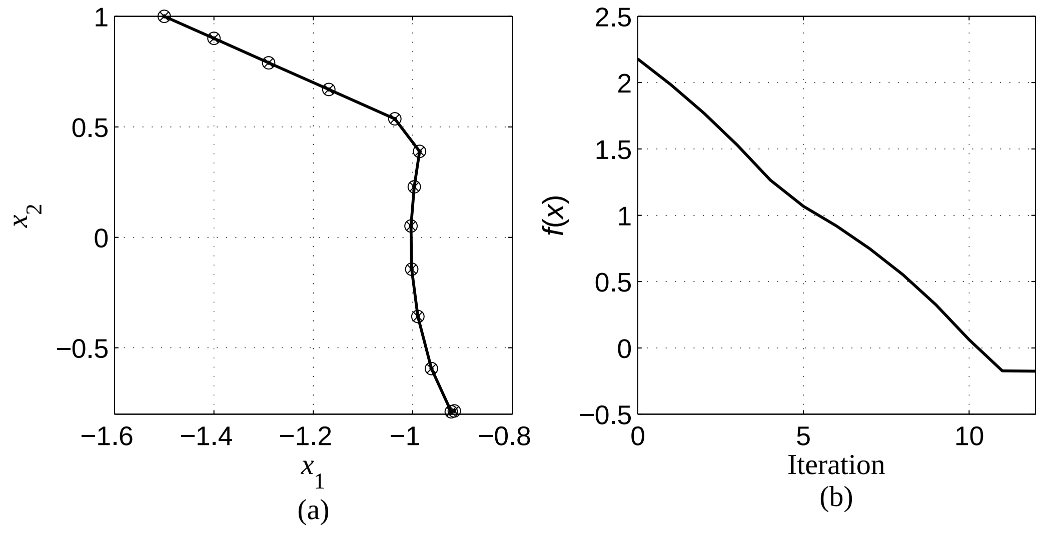


Figure 5.2. (a) First twelve iterations produced by Algorithm 5.3, (b) objective function versus iteration for Example 5.2.

5.2.4 Alternating convex optimization

The method of alternating convex optimization [1] to be studied in this section can be explained as follows. Suppose the number of variables in problem (5.1) is n. We divide index set {1, 2, …,

$n\}$ into k non-overlapping subsets I_i such that

$$\bigcup_{i=1}^{k} I_i = \{1, 2, \ldots, n\} \tag{5.11}$$

The components of variable $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T$, can then be divided into k groups accordingly, with each group forming a vector which we denote by $\boldsymbol{x}^{(i)}$ for i = 1, 2, …, k. In other words the variable vector x is now arranged be collecting these *sub-variables* $\{\boldsymbol{x}^{(i)}$ for i = 1, 2, …, k $\}$ as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{(1)}; & \boldsymbol{x}^{(2)}; & \cdots & \boldsymbol{x}^{(k)} \end{bmatrix}$$

We assume that there exists an index division in Eq. (5.11) such that if we consider any specific variable, say $\boldsymbol{x}^{(i)}$, as the only variable with the rest of the variables fixed, the problem in Eq. (5.1) is *convex with respect to* $\boldsymbol{x}^{(i)}$. Such an index division $\{I_i$, for i = 1, 2, …, $k\}$ is said to be *regular* for problem (5.1). A natural SCP strategy under this circumstance is that in each step the problem in Eq. (5.1) is solved with respect to one specific variable, say $\boldsymbol{x}^{(i)}$, with the rest of the variables fixed so that the problem becomes convex; and the procedure is repeated with i cycling through enough number of times until variations in all variables $\{\boldsymbol{x}^{(i)}$ for i = 1, 2, …, k $\}$ or in the objective function or in both are less than a given tolerance. Evidently, the values of the objective function at the iterates produced by this method are always non-increasing. On the other hand, due to the nonconvex nature of problem (5.1) the method is not guaranteed to converge to the global minimizer, and the performance of the local solution the algorithm converges to depends to a large extent how the algorithm is initiated. We now summarized the method as an SCP algorithm below.

Algorithm 5.4 Alternating convex optimization for problem (5.1)

Step 1 Input a regular index division $\{I_i$, for i = 1, 2, …, $k\}$, a tolerance ε > 0, an initial point $\boldsymbol{x}_0$ arranged in accordance with the index division as

$\boldsymbol{x}_0 = \begin{bmatrix} \boldsymbol{x}_0^{(1)}; & \boldsymbol{x}_0^{(2)}; & \cdots & \boldsymbol{x}_0^{(k)} \end{bmatrix}$. Set l = 0.

Step 2 For i = 1, 2. …, k, with $\boldsymbol{x}_l$ as initial point solve the problem in Eq. (5.1) with respect to variable $\boldsymbol{x}^{(i)}$ by using an appropriate convex solver. Denote the solution obtained by $\boldsymbol{x}_{l+1}$.

Step 3 If $\left|\frac{f(\boldsymbol{x}_{l+1}) - f(\boldsymbol{x}_l)}{f(\boldsymbol{x}_l)}\right| < \varepsilon$, stop and output $\boldsymbol{x}_{l+1}$ as solution; otherwise set $l = l + 1$ and repeat from Step 2.

Example 5.3 Find nonnegative matrixes $\boldsymbol{X}$ and $\boldsymbol{Y}$ of sizes m by r and r by n, respectively, that solve the problem

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{X},\boldsymbol{Y}) = \tfrac{1}{2}\|\boldsymbol{XY} - \boldsymbol{D}\|_F^2 \\ \text{subject to:} & \boldsymbol{X} \ge \boldsymbol{0},\ \boldsymbol{Y} \ge \boldsymbol{0} \end{array} \tag{5.12a-b}$$

where $\boldsymbol{D}$ is a given data matrix of size m by n and $r \ll \min\{m, n\}$. The constraints in Eq. (5.12b) are imposed in component-wise sense, meaning that every component of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is constrained to be nonnegative. In the literature, the problem is known as the *nonnegative matrix factorization* problem which finds applications in image processing. Illustrate the solution method by applying it to a numerical instance where matrix $\boldsymbol{D}$ is a 64 by 64 portion of a test image, known as cameraman, see Fig. 5.5(a). The component values of $\boldsymbol{D}$ are normalized to within the unit interval [0, 1]. Parameter r is set to 15.

Solution

From Eq. (5.12a), we see that if we fix $\boldsymbol{X}$ ($\boldsymbol{Y}$), the objective function is quadratic and convex with respect to $\boldsymbol{Y}$ ($\boldsymbol{X}$). Since the constraints in Eq. (5.12b) are linear, with either $\boldsymbol{X}$ or $\boldsymbol{Y}$ fixed the problem in Eq. (5.12) is convex, thus it can be tackled by Algorithm 5.4. Since there are only two sub-variables, namely $\boldsymbol{x}^{(1)} = \boldsymbol{X}$ and $\boldsymbol{x}^{(2)} = \boldsymbol{Y}$, Step 2 of Algorithm 5.4 is simplified to solving two convex similar QP problems in an alternating manner as detailed below. We start with a random nonnegative matrix of size 15 by 64 as initial $\boldsymbol{Y}$. With $\boldsymbol{Y}$ held fixed, the problem in Eq. (5.12) is a convex QP problem with respect to $\boldsymbol{X}$. The solution of this QP problem, $\boldsymbol{X}$, is then held fixed and the problem in Eq. (5.12) is again a convex QP problem with respect to $\boldsymbol{Y}$. The procedure is repeated until the relative error in objective function is less than $\varepsilon = 10^{-4}$. Fig. 5.3 shows profiles of the objective function versus iteration number for four randomly generated initial points. It is observed that different initial points do lead to different local solutions. Fig. 5.4 shows one of the local solutions $\{\boldsymbol{X}^*,\boldsymbol{Y}^*\}$. In image processing context, the columns of $\boldsymbol{X}^*$ are interpreted as the *features* of the data matrix $\boldsymbol{D}$ and the components in each column of $\boldsymbol{Y}^*$ are weights that linearly combines the image features to represent the data image. Note that both $\boldsymbol{X}^*$ and $\boldsymbol{Y}^*$ contain many black spots which correspond to near zero numerical values, hence both matrices $\boldsymbol{X}^*$ and $\boldsymbol{Y}^*$ are considered sparse. This implies that a typical image can be effectively represented by linearly combining a small number of sparse image features. Fig. 5.5(b) shows the product $\boldsymbol{X}^*\boldsymbol{Y}^*$ as an image approximating matrix $\boldsymbol{D}$. The value of the objective function at solution $\{\boldsymbol{X}^*,\boldsymbol{Y}^*\}$ is found to be $f(\boldsymbol{X}^*,\boldsymbol{Y}^*) = 10.4504$. ■

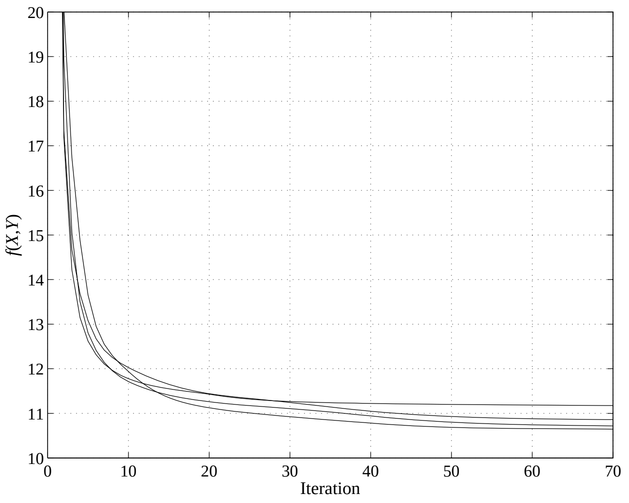


Figure 5.3. Objective function versus iteration with 4 randomly generated initial points.

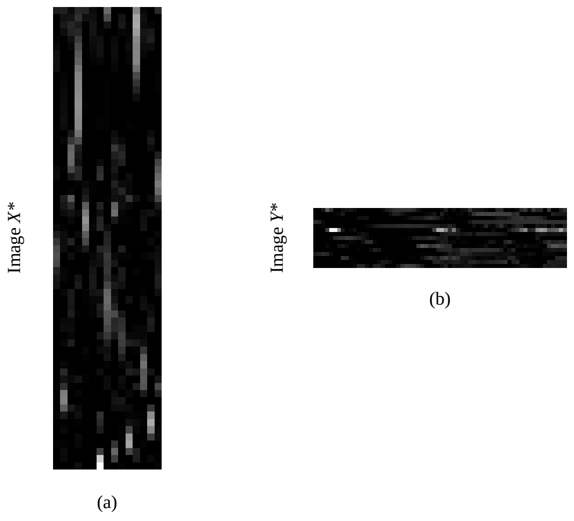


Figure 5.4. A local solution $\{\boldsymbol{X}^*, \boldsymbol{Y}^*\}$. Note that the columns of $\boldsymbol{X}^*$ as image features are sparse, and the components of $\boldsymbol{Y}^*$ as weights are also sparse.

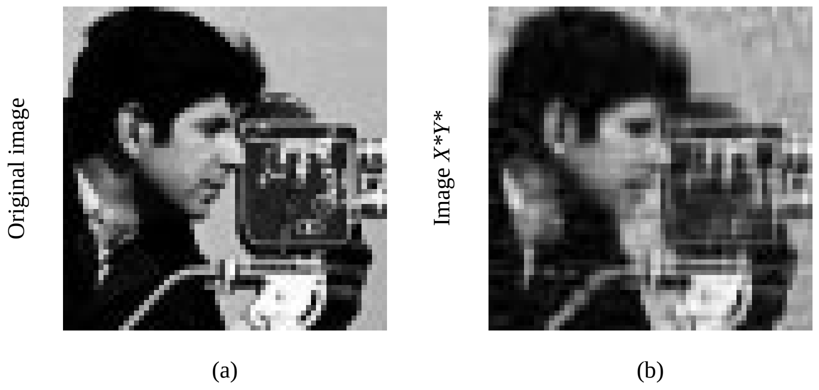


Figure 5.5. (a) Original patch from image cameraman, (b) $\boldsymbol{X}^*\boldsymbol{Y}^*$ as an image.

Example 5.4 This example illustrates the alternating convex optimization method by applying it to deal with nonconvex problems encountered in *recommender systems*. The material below is largely based on the well-known 2009 paper by Koren, Bell, and Volinsky:

Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," ***Computer***, **pp. 30-37, August 2009.**

System Model

We begin by quoting the above reference: "Modern consumer are inundated with choices. …… Matching consumers with the most appropriate product is the key to enhancing user satisfaction and loyalty. Therefore, more retailers have become interested in recommendation systems, which analyze patterns of user interest in products to provide personalized recommendations that suit a user's taste. …… Such systems are particularly useful for entertainment products such as movies, music, and TV shows. Many customers will view the same movie, and each customer is likely to view numerous different movies. Customers have proven willing to indicate their level of satisfaction with particular movies, so huge volume of data is available about which movies appeal to which customers. Companies can analyze this data to recommend movies to particular customers."

There are two strategies based on which recommender systems are developed, namely the *content filtering* and *collaborative filtering* approaches. "The content filtering approach creates a profile for each user or product to characterize its nature", and the profiles associate users with matching products. As a result, "content-based strategies require gathering external information that might not be available or easy to collect". On the other hand, the collaborative filtering "relies only on past user behavior – for example, previous transactions or product ratings – without requiring the creation of explicit profiles." In this example, we are interested in models from the collaborative filtering approach, known as *latent factor* models, especially their realizations based on *matrix factorization*.

To describe a basic matrix factorization model, we continue to quote the above reference: "Recommender systems rely on different types of input data, which are often placed in a matrix with one dimension representing users and the other dimension representing items of interest. The most convenient data is high-quality *explicit feedback*, which includes explicit input by users regarding their interest in products. For example, **Netflix** collects star ratings for movies, and **TiVo** users indicate their preferences for TV shows by pressing thumbs-up and thumbs-down buttons. We refer to explicit user feedback as *ratings*. Usually, explicit feedback comprises a sparse matrix, since any single user is likely to have rated only a small percentage of possible items."

Matrix factorization models map both users and items to a joint *latent factor* space of dimensionality *d*, such that user-item interactions are modeled as inner products in that space. Accordingly, each user *i* is associated with a vector $\boldsymbol{u}_i \in R^d$, and each item *j* is associated with a vector $\boldsymbol{v}_j \in R^d$. For a given user *i*, the components of $\boldsymbol{u}_i$ measure the extent of interest the user has in items that are high on the corresponding factors, positive or negative. For a given item *j*, the components of $\boldsymbol{v}_j$ measure the extent to which the item possesses those factors, again, positive or negative. The inner product, $\boldsymbol{u}_i^T \boldsymbol{v}_j$, captures the interaction between user *i* and item *j*

– the user's overall interest in the item's characteristics. This approximates user *i*'s rating of item *j*, which is denoted by $r_{i,j}$, leading to an estimate

$$\hat{r}_{i,j} = \boldsymbol{u}_i^T \boldsymbol{v}_j$$

As noted in the above reference, in addition, "much of the observed variation in rating values is due to effects associated with either users or items, known as biases or intercepts, independent of any interactions. For example, typical collaborative filtering data exhibits large systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others. After all, some products are widely perceived as better (or worse) than others." A first-order approximation of the bias involved in rating $r_{i,j}$ is given by

$$b_{i,j} = b_i^{(1)} + b_j^{(2)} + \mu$$

where $b_i^{(1)}$ and $b_j^{(2)}$ denote the deviations of user *i* and item *j* from the average, respectively, and μ denotes the overall average rating. By taking both user-product interaction as well as user and item biases into account, a better rating model is obtained as

$$\hat{r}_{i,j} = \boldsymbol{u}_i^T \boldsymbol{v}_j + b_i^{(1)} + b_j^{(2)} + \mu \qquad (5.13)$$

Problem Formulation

The major challenge encountered in the application of rating model (5.13) is computing the mapping of each item and user to factor vectors $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$, as well as biases $b_i^{(1)}$ and $b_j^{(2)}$. More recent works suggest modeling directly the observed ratings only, while avoiding overfitting through a *regularized* model. Specifically, the model parameters in (5.13) are determined by minimizing the squared error function

$$\underset{\{\boldsymbol{u}_i, b_i^{(1)}, \boldsymbol{v}_j, b_j^{(2)}\}}{\text{minimize}} \; f(\boldsymbol{x}) = \frac{1}{2} \sum_{\{i,j\} \in \mathcal{K}} \left(\boldsymbol{u}_i^T \boldsymbol{v}_j + b_i^{(1)} + b_j^{(2)} + \mu - r_{i,j} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{m} \left(\| \boldsymbol{u}_i \|_2^2 + b_i^{(1)2} \right) + \frac{\lambda}{2} \sum_{j=1}^{n} \left(\| \boldsymbol{v}_j \|_2^2 + b_j^{(2)2} \right)$$

(5.14)

where $\boldsymbol{x}$ collects of all unknown parameters, i.e. $\boldsymbol{x} = \{\boldsymbol{u}_i, b_i^{(1)}, \boldsymbol{v}_j, b_j^{(2)}, \text{for } i = 1, 2, \ldots, m; j = 1, 2, \ldots, n\}$ involving a total of *m* users and *n* items, $\mathcal{K}$ denotes the set of (*i, j*) pairs for which rating $r_{i,j}$ is *known*, and $\lambda > 0$ is a regularization parameter.

***Solving Problem* (5.14)**

Obviously, with respect to model parameter $\boldsymbol{x}$ the problem in (5.14) is non-quadratic and nonconvex. Nevertheless, its objective has a structure that is naturally suitable for alternating convex optimization. In effect if we let

$$\boldsymbol{x}_1 = \begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_m \\ b_1^{(1)} \\ \vdots \\ b_m^{(1)} \end{bmatrix}, \quad \text{and} \quad \boldsymbol{x}_2 = \begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_n \\ b_1^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix} \tag{5.15}$$

then, with $\boldsymbol{x}_2$ fixed $f(\boldsymbol{x})$ is quadratic and convex with respect to $\boldsymbol{x}_1$; and with $\boldsymbol{x}_1$ fixed $f(\boldsymbol{x})$ is quadratic and convex with respect to $\boldsymbol{x}_2$. From (5.14) and (5.15) it follows that

$$\nabla_{\boldsymbol{x}_1} f(\boldsymbol{x}) = \begin{bmatrix} \nabla_{\boldsymbol{u}_1} f(\boldsymbol{x}) \\ \vdots \\ \nabla_{\boldsymbol{u}_m} f(\boldsymbol{x}) \\ \nabla_{b_1^{(1)}} f(\boldsymbol{x}) \\ \vdots \\ \nabla_{b_m^{(1)}} f(\boldsymbol{x}) \end{bmatrix} \quad \text{and} \quad \nabla_{\boldsymbol{x}_2} f(\boldsymbol{x}) = \begin{bmatrix} \nabla_{\boldsymbol{v}_1} f(\boldsymbol{x}) \\ \vdots \\ \nabla_{\boldsymbol{v}_n} f(\boldsymbol{x}) \\ \nabla_{b_1^{(2)}} f(\boldsymbol{x}) \\ \vdots \\ \nabla_{b_n^{(2)}} f(\boldsymbol{x}) \end{bmatrix}$$

To compute $\nabla_{\boldsymbol{u}_i} f(\boldsymbol{x})$ and $\nabla_{b_i^{(1)}} f(\boldsymbol{x})$, we divide set $\mathcal{K}$ into *m* non-overlapping subsets, that is $\mathcal{K} = \bigcup_{i=1}^{m} \mathcal{S}_i$, where each $\mathcal{S}_i$ collects those (i, j) pairs from set $\mathcal{K}$ with the same index *i*. Now the objective in (5.14) can be expressed as

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{m} \sum_{\{i,j\}\in \mathcal{S}_i} \left(\boldsymbol{u}_i^T \boldsymbol{v}_j + b_i^{(1)} + b_j^{(2)} + \mu - r_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\left(\|\boldsymbol{u}_i\|_2^2 + b_i^{(1)\,2}\right) + \frac{\lambda}{2}\sum_{j=1}^{n}\left(\|\boldsymbol{v}_j\|_2^2 + b_j^{(2)\,2}\right)$$

Therefore, for *i* = 1, 2, …, *m*,

$$\nabla_{\boldsymbol{u}_i} f(\boldsymbol{x}) = \lambda \boldsymbol{u}_i + \sum_{\{i,j\}\in S_i} \left(\boldsymbol{u}_i^T \boldsymbol{v}_j + b_i^{(1)} + b_j^{(2)} + \mu - r_{i,j}\right)\boldsymbol{v}_j \tag{5.16a}$$

and

$$\nabla_{b_i^{(1)}} f(\boldsymbol{x}) = (\lambda + |\mathcal{S}_i|)\, b_i^{(1)} + \sum_{\{i,j\}\in S_i} \left(\boldsymbol{u}_i^T \boldsymbol{v}_j + b_j^{(2)} + \mu - r_{i,j}\right) \tag{5.16b}$$

where $|\mathcal{S}_i|$ denotes the cardinality of set $\mathcal{S}_i$. Similarly, by dividing set $\mathcal{K}$ into *n* non-overlapping subsets, that is $\mathcal{K} = \bigcup_{j=1}^{n} \mathcal{T}_j$, where each $\mathcal{T}_j$ collects those (i, j) pairs from set $\mathcal{K}$ with the same index *j*, the objective in (5.14) can be expressed as

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{j=1}^{n} \sum_{\{i,j\}\in \mathcal{T}_j} \left(\boldsymbol{u}_i^T \boldsymbol{v}_j + b_i^{(1)} + b_j^{(2)} + \mu - r_{i,j}\right)^2 + \frac{\lambda}{2}\sum_{i=1}^{m}\left(\|\boldsymbol{u}_i\|_2^2 + b_i^{(1)\,2}\right) + \frac{\lambda}{2}\sum_{j=1}^{n}\left(\|\boldsymbol{v}_j\|_2^2 + b_j^{(2)\,2}\right)$$

Thus for *j* = 1, 2, …, *n*,

$$\nabla_{\boldsymbol{v}_j} f(\boldsymbol{x}) = \lambda \boldsymbol{v}_j + \sum_{\{i,j\}\in T_j} \left(\boldsymbol{u}_i^T \boldsymbol{v}_j + b_i^{(1)} + b_j^{(2)} + \mu - r_{i,j}\right)\boldsymbol{u}_i \tag{5.17a}$$

and

$$\nabla_{b_j^{(2)}} f(\boldsymbol{x}) = (\lambda + |\mathcal{T}_j|) b_j^{(2)} + \sum_{\{i,j\}\in\mathcal{T}_j} \left(\boldsymbol{u}_i^T \boldsymbol{v}_j + b_i^{(1)} + \mu - r_{i,j} \right) \tag{5.17b}$$

In a single (say, the *k*th) round of alternating minimization (AM), we minimize the objective function w.r.t. $\boldsymbol{x}_1$ with variable $\boldsymbol{x}_2$ fixed by solving the linear system of equations $\nabla_{\boldsymbol{x}_1} f(\boldsymbol{x}) = \boldsymbol{0}$. These linear equations can be specified using (5.16) as

$$\begin{bmatrix} \boldsymbol{\Gamma}_1^{(2)} & \cdots & \boldsymbol{0} & \boldsymbol{p}_1^{(2)} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{\Gamma}_m^{(2)} & \boldsymbol{0} & \cdots & \boldsymbol{p}_m^{(2)} \\ \boldsymbol{p}_1^{(2)T} & \cdots & \boldsymbol{0} & \delta_1^{(2)} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{p}_m^{(2)T} & 0 & \cdots & \delta_m^{(2)} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_m \\ b_1^{(1)} \\ \vdots \\ b_m^{(1)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_1^{(2)} \\ \vdots \\ \boldsymbol{q}_m^{(2)} \\ \tau_1^{(2)} \\ \vdots \\ \tau_m^{(2)} \end{bmatrix} \tag{5.18}$$

where

$\boldsymbol{\Gamma}_i^{(2)} = \lambda \boldsymbol{I} + \sum_{\{i,j\}\in\mathcal{S}_i} \boldsymbol{v}_j \boldsymbol{v}_j^T$, $\boldsymbol{p}_i^{(2)} = \sum_{\{i,j\}\in\mathcal{S}_i} \boldsymbol{v}_j$, $\boldsymbol{q}_i^{(2)} = \sum_{\{i,j\}\in\mathcal{S}_i} \left(r_{i,j} - b_j^{(2)} - \mu \right) \boldsymbol{v}_j$, $\delta_i^{(2)} = \lambda + |\mathcal{S}_i|$, and $\tau_i^{(2)} = \sum_{\{i,j\}\in\mathcal{S}_i} \left(r_{i,j} - b_j^{(2)} - \mu \right)$.

Once the solution of (5.18), denoted by $\boldsymbol{x}_1^{(k)}$, is obtained, next we fix $\boldsymbol{x}_1$ to $\boldsymbol{x}_1 = \boldsymbol{x}_1^{(k)}$ and minimize the objective function w.r.t. $\boldsymbol{x}_2$ by solving the linear system of equations $\nabla_{\boldsymbol{x}_2} f(\boldsymbol{x}) = \boldsymbol{0}$. Using (5.17) these equations can be specified as

$$\begin{bmatrix} \boldsymbol{\Gamma}_1^{(1)} & \cdots & \boldsymbol{0} & \boldsymbol{p}_1^{(1)} & \cdots & \boldsymbol{0} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{\Gamma}_n^{(1)} & \boldsymbol{0} & \cdots & \boldsymbol{p}_n^{(1)} \\ \boldsymbol{p}_1^{(1)T} & \cdots & \boldsymbol{0} & \delta_1^{(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{p}_n^{(1)T} & 0 & \cdots & \delta_n^{(1)} \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1 \\ \vdots \\ \boldsymbol{v}_n \\ b_1^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_1^{(1)} \\ \vdots \\ \boldsymbol{q}_n^{(1)} \\ \tau_1^{(1)} \\ \vdots \\ \tau_n^{(1)} \end{bmatrix} \tag{5.19}$$

where

$\boldsymbol{\Gamma}_j^{(1)} = \lambda \boldsymbol{I} + \sum_{\{i,j\}\in\mathcal{T}_j} \boldsymbol{u}_i \boldsymbol{u}_i^T$, $\boldsymbol{p}_j^{(1)} = \sum_{\{i,j\}\in\mathcal{T}_j} \boldsymbol{u}_i$, $\boldsymbol{q}_j^{(1)} = \sum_{\{i,j\}\in\mathcal{T}_j} \left(r_{i,j} - b_i^{(1)} - \mu \right) \boldsymbol{u}_i$, $\delta_j^{(1)} = \lambda + |\mathcal{T}_j|$, and $\tau_j^{(1)} = \sum_{\{i,j\}\in\mathcal{T}_j} \left(r_{i,j} - b_i^{(1)} - \mu \right)$. We denote the solution of (5.19) by $\boldsymbol{x}_2^{(k)}$, and the *k*th round of AM is now complete.

It is important to remark that when the AM algorithm starts with an initial $\boldsymbol{x}_2^{(0)}$, each round of AM always reduces the (original) objective function, hence the algorithm converges due to monotonic decrease of the objective. However, different initial $\boldsymbol{x}_2^{(0)}$ will likely lead to different solutions because of the non-convexity of the original problem.

The AM algorithm was tested by a numerical example of star-rating movies involving three movies (items) and three viewers (users), with 7 known ratings as shown in the table below:

Viewer\Movie	**Doctor Strange**	**Star Trek: Beyond**	**Zootopia**
Alice	1	?	5
Dave	3	4	?
Joe	3	5	2

Based on the given data, the problem is to best estimate the ratings of Star Trek by Alice and of Zootopia by Dave. This example is of a small-size with $m = n = 3$. The dimension of the latent factors was chosen to be $d = 3$. The overall average rating can be estimated by the mean of the known ratings as $\mu \approx 23/7 = 3.2857$. The initial $\boldsymbol{v}_i$'s and $b_i^{(2)}$'s were set randomly as follows:

```
randn('state',17);
r = 0.01*randn(3,4);
v1 = r(:,1); v2 = r(:,2); v3 = r(:,3);
b21 = r(1,4); b22 = r(2,4); b23 = r(3,4);
```

With $\lambda = 0.01$, 400 rounds of AM reduces the objective function from $f(\boldsymbol{x}^{(0)}) = 6.3704$ to a steady-state error $f(\boldsymbol{x}^{(400)}) = 4.0995 \times 10^{-2}$. Using the rating model in (5.13) with optimized latent factors and biases, optimal rating estimates $\{\hat{r}_{i,j}, i, j = 1,2,3\}$ are found to be

```
    1.007581807818565   2.878615874233677   4.994229635010283
    2.996670833299223   4.001670055582038   3.368475590121493
    3.002929001656147   4.992182680298524   2.004936810934420
```

which show good matches between the seven known ratings and their estimates. Our estimates of the ratings of both Star Trek by Alice and of Zootopia by Dave are 3 stars. We remark that the results appear to be quite robust against small changes in initial point, regularization parameter λ, and average rating μ. ■

5.3 Sequential Quadratic Programming

In this section we explore a class of algorithms, known as *sequential quadratic programming* (SQP) algorithms [2][3], for the general constrained optimization problem in Eq. (5.1) which we present below for convenient reference:

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to:} \;\; a_i(\boldsymbol{x}) = 0 \;\; \text{for } i = 1,2,\ldots,p \\ &\qquad\qquad\quad c_j(\boldsymbol{x}) \le 0 \;\; \text{for } j = 1,2,\ldots,q \end{aligned}$$

where functions $f(\boldsymbol{x})$, $a_i(\boldsymbol{x})$ for i = 1, 2, …, p and $c_j(\boldsymbol{x})$ for j = 1, 2, …, q are twice continuously differentiable.

5.3.1 Basic SQP algorithm

The basic SQP algorithm is in spirit similar to the SCP algorithms studied in Sec. 5.2, but it possesses several distinct features. We start by recalling the KKT conditions for the problem in Eq. (5.1), which are given by

$$\begin{aligned} \nabla_x L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}) &= \mathbf{0} \\ a_i(\boldsymbol{x}) &= 0 \quad \text{for } i = 1,2,\ldots,p \\ c_j(\boldsymbol{x}) &\le 0 \quad \text{for } j = 1,2,\ldots,q \\ \boldsymbol{\mu} &\ge \mathbf{0} \\ \mu_j c_j(\boldsymbol{x}) &= 0 \quad \text{for } j = 1,2,\ldots,q \end{aligned} \tag{5.20a-e}$$

where the Lagrangian assumes the form

$$L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i a_i(\boldsymbol{x}) + \sum_{j=1}^{q} \mu_j c_j(\boldsymbol{x})$$

In the *k*th iteration, we update a known iterate $\{\boldsymbol{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k\}$ to a new iterate $\{\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}, \boldsymbol{\mu}_{k+1}\}$ so that the KKT conditions in Eq. (5.20) are satisfied at $\{\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}, \boldsymbol{\mu}_{k+1}\}$ approximately. To this end, we let $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{\delta}_x$ and examine the affine approximation of Eq. (5.20) at $\{\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}, \boldsymbol{\mu}_{k+1}\}$, which are given by

$$\begin{aligned} \boldsymbol{Z}_k \boldsymbol{\delta}_x + \boldsymbol{g}_k + \boldsymbol{A}_{ek}^T \boldsymbol{\lambda}_{k+1} + \boldsymbol{A}_{ik}^T \boldsymbol{\mu}_{k+1} &= \mathbf{0} \\ \boldsymbol{A}_{ek} \boldsymbol{\delta}_x &= -\boldsymbol{a}_k \\ \boldsymbol{A}_{ik} \boldsymbol{\delta}_x &\le -\boldsymbol{c}_k \\ \boldsymbol{\mu}_{k+1} &\ge \mathbf{0} \\ (\boldsymbol{\mu}_{k+1})_j (\boldsymbol{A}_{ik} \boldsymbol{\delta}_x + \boldsymbol{c}_k)_j &= 0 \qquad \text{for } j = 1,2,\ldots,q \end{aligned} \tag{5.21a-e}$$

where

$$\boldsymbol{Z}_k = \nabla_x^2 f(\boldsymbol{x}_k) + \sum_{i=1}^{p} (\boldsymbol{\lambda}_k)_i \nabla_x^2 a_i(\boldsymbol{x}_k) + \sum_{j=1}^{q} (\boldsymbol{\mu}_k)_j \nabla_x^2 c_j(\boldsymbol{x}_k) \tag{5.22a}$$

$$\boldsymbol{g}_k = \nabla_x f(\boldsymbol{x}_k) \tag{5.22b}$$

$$\boldsymbol{A}_{ek} = \begin{bmatrix} \nabla_x^T a_1(\boldsymbol{x}_k) \\ \nabla_x^T a_2(\boldsymbol{x}_k) \\ \vdots \\ \nabla_x^T a_p(\boldsymbol{x}_k) \end{bmatrix}, \quad \boldsymbol{A}_{ik} = \begin{bmatrix} \nabla_x^T c_1(\boldsymbol{x}_k) \\ \nabla_x^T c_2(\boldsymbol{x}_k) \\ \vdots \\ \nabla_x^T c_q(\boldsymbol{x}_k) \end{bmatrix}, \quad \boldsymbol{a}_k = \begin{bmatrix} a_1(\boldsymbol{x}_k) \\ a_2(\boldsymbol{x}_k) \\ \vdots \\ a_p(\boldsymbol{x}_k) \end{bmatrix}, \quad \boldsymbol{c}_k = \begin{bmatrix} c_1(\boldsymbol{x}_k) \\ c_2(\boldsymbol{x}_k) \\ \vdots \\ c_q(\boldsymbol{x}_k) \end{bmatrix} \tag{5.22c}$$

Eq. (5.21) may be interpreted as the exact KKT conditions of the QP problem

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\boldsymbol{\delta}^T \boldsymbol{Z}_k \boldsymbol{\delta} + \boldsymbol{\delta}^T \boldsymbol{g}_k \\ \text{subject to:} \quad & \boldsymbol{A}_{ek}\boldsymbol{\delta} = -\boldsymbol{a}_k \\ & \boldsymbol{A}_{ik}\boldsymbol{\delta} \le -\boldsymbol{c}_k \end{aligned} \tag{5.23a-c}$$

Let $\boldsymbol{\delta}_x$ be the solution of the above QP problem and $\{\boldsymbol{\lambda}_{qp}, \boldsymbol{\mu}_{qp}\}$ be the associated Lagrange multipliers. If we define

$$\boldsymbol{\delta}_\lambda = \boldsymbol{\lambda}_{qp} - \boldsymbol{\lambda}_k \text{ and } \boldsymbol{\delta}_\mu = \boldsymbol{\mu}_{qp} - \boldsymbol{\mu}_k \tag{5.24}$$

the new iterate $\{\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}, \boldsymbol{\mu}_{k+1}\}$ can be obtained as

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{\delta}_x \tag{5.25a}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha_k \boldsymbol{\delta}_\lambda \tag{5.25b}$$

$$\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \alpha_k \boldsymbol{\delta}_\mu \tag{5.25c}$$

where α_k is determined by a line search procedure that involves minimizing a *merit function*. Popular merit functions in the context of SQP include the l_1 merit function [4], [5] and the augmented Lagrangian merit function [6]-[8]. The l_1 merit function is defined by

$$\psi_k(\alpha) = f(\boldsymbol{x}_k + \alpha\boldsymbol{\delta}_x) + \tau\sum_{i=1}^{p} |a_i(\boldsymbol{x}_k + \alpha\boldsymbol{\delta}_x)| + \tau\sum_{j=1}^{q} c_j(\boldsymbol{x}_k + \alpha\boldsymbol{\delta}_x)_+ \tag{5.26}$$

where $c_j(\boldsymbol{x}_k + \alpha\boldsymbol{\delta}_x)_+ = \max\{c_j(\boldsymbol{x}_k + \alpha\boldsymbol{\delta}_x), 0\}$ and $\tau > 0$ is a sufficiently large scalar. Evidently, minimizing a merit function with respect to α reduces the objective function along the search direction $\boldsymbol{\delta}_x$ and, at the same time, reduces the degree of violation for both the equality and inequality constraints. Therefore a good choice of α_k for Eq. (5.25) is obtained by minimizing $\psi_k(\alpha)$ over a reasonable interval, say, $0 \le \alpha \le 1$, namely,

$$\alpha_k = \arg\min_{0\le\alpha\le1} \psi_k(\alpha)$$

Because of the presence of the second and third terms in Eq. (5.26), the merit function is non-differential in general and is likely nonconvex. A straightforward way to obtain a good estimate of α_k, which is found quite effective especially when the number of constraints is not large, is to evaluate $\psi_k(\alpha)$ over a set of discrete grid points $\mathcal{A} = \{\alpha_l, l = 1, 2, ..., L\}$ that are placed uniformly on the unit interval and identify a grid point at which $\psi_k(\alpha)$ assumes smallest value:

$$\alpha_k \approx \arg\min_{\alpha\in\mathcal{A}} \psi_k(\alpha) \tag{5.27}$$

The basic SQP algorithm can now be summarized as follows.

Algorithm 5.5 Basic SQP algorithm for problem (5.1)

Step 1 Input initial point $\{\boldsymbol{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0\}$, $\tau > 0$, and tolerance ε. Set $k = 0$.

Step 2 Solve the QP problem in Eq. (5.23) to obtain $\boldsymbol{\delta}_x$ and $\{\boldsymbol{\lambda}_{qp}, \boldsymbol{\mu}_{qp}\}$, then using Eq. (5.24) to obtain $\boldsymbol{\delta}_\lambda$ and $\boldsymbol{\delta}_\mu$.

Step 3 Choose α_k using Eq. (5.27).

Step 4 Set $\{\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}, \boldsymbol{\mu}_{k+1}\}$ using Eq. (5.25).

Step 5 If $\|[\boldsymbol{x}_{k+1} - \boldsymbol{x}_k;\ \boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k;\ \boldsymbol{\mu}_{k+1} - \boldsymbol{\mu}_k]\|_2 < \varepsilon$, stop and output $\boldsymbol{x}_{k+1}$ as solution; otherwise set $k = k + 1$ and repeat from Step 2.

5.3.2 Positive definite approximation of Hessian $\boldsymbol{Z}_k$

As can be seen from Eq. (5.22a), computing matrix $\boldsymbol{Z}_k$ is quite involved and, more importantly, $\boldsymbol{Z}_k$ is not guaranteed to be positive definite or positive semidefinite, especially when the current iterate $\boldsymbol{x}_k$ is not sufficiently close to a solution point $\boldsymbol{x}^*$. This can in turn cause difficulties in solving the QP subproblem in Eq. (5.16). One way to handle this problem is to use an approximate Hessian, denoted again by $\boldsymbol{Z}_k$, which is updated using a modified BFGS formula [9] as

$$\boldsymbol{Z}_{k+1} = \boldsymbol{Z}_k + \frac{\boldsymbol{\eta}_k\boldsymbol{\eta}_k^T}{\boldsymbol{\delta}_x^T\boldsymbol{\eta}_k} - \frac{\boldsymbol{Z}_k\boldsymbol{\delta}_x\boldsymbol{\delta}_x^T\boldsymbol{Z}_k}{\boldsymbol{\delta}_x^T\boldsymbol{Z}_k\boldsymbol{\delta}_x} \tag{5.28a}$$

where

$$\begin{aligned}
\boldsymbol{\eta}_k &= \theta\boldsymbol{\gamma}_k + (1-\theta)\boldsymbol{Z}_k\boldsymbol{\delta}_x \\
\boldsymbol{\gamma}_k &= (\boldsymbol{g}_{k+1} - \boldsymbol{g}_k) + (\boldsymbol{A}_{e,k+1} - \boldsymbol{A}_{e,k})^T\boldsymbol{\lambda}_{k+1} + (\boldsymbol{A}_{i,k+1} - \boldsymbol{A}_{i,k})^T\boldsymbol{\mu}_{k+1} \\
\theta &= \begin{cases} 1 & \text{if } \boldsymbol{\delta}_x^T\boldsymbol{\gamma}_k \ge 0.2\boldsymbol{\delta}_x^T\boldsymbol{Z}_k\boldsymbol{\delta}_x \\ \dfrac{0.8\boldsymbol{\delta}_x^T\boldsymbol{Z}_k\boldsymbol{\delta}_x}{\boldsymbol{\delta}_x^T\boldsymbol{Z}_k\boldsymbol{\delta}_x - \boldsymbol{\delta}_x^T\boldsymbol{\gamma}_k} & \text{otherwise} \end{cases}
\end{aligned} \tag{5.28b-d}$$

and the initial $\boldsymbol{Z}_0$ is usually set to $\boldsymbol{I}_n$. A desirable property of above BFGS approximation is that

$\boldsymbol{Z}_{k+1}$ is positive definite if $\boldsymbol{Z}_k$ is positive definite. As a result, the QP problem in Eq. (5.23) where $\boldsymbol{Z}_k$ is generated by Eq. (5.28) is a convex QP problem.

5.3.3 Robustness and solvability of QP subproblem

It is important to stress that the development of Algorithm 5.5 is based on the affine approximation of the KKT conditions in Eq. (5.21), and the validity of such approximation is based on the assumption that the variable $\boldsymbol{\delta}_x$ is sufficiently small in magnitude. In many practical problems, including a bound constraint such as $\|\boldsymbol{\delta}_x\|_2 \le \rho$ is found to make the algorithm more robust in dealing with the technical difficulties due to the use of an initial point that is far from the solution point.

Another technical issue that may occur in implementing an SQP algorithm is that the QP subproblem in Eq. (5.23) may be unsolvable even if matrix $\boldsymbol{Z}_k$ in Eq. (5.23a) is positive definite. This will be indeed the case when the feasible region defined by Eqs. (5.23b) and (5.23c) turns out to be empty. This problem can be fixed by introducing nonnegative slack variables $\boldsymbol{u}, \boldsymbol{v} \in R^p$, $\boldsymbol{w} \in R^q$, and $r \in R$ to reformulate the QP subproblem as

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\boldsymbol{\delta}^T \boldsymbol{Z}_k \boldsymbol{\delta} + \boldsymbol{\delta}^T \boldsymbol{g}_k + \eta\left(r + \sum_{i=1}^{p}(u_i + v_i) + \sum_{j=1}^{p} w_j \right) \\ \text{subject to:} \quad & \boldsymbol{A}_{ek}\boldsymbol{\delta} + \boldsymbol{a}_k = \boldsymbol{u} - \boldsymbol{v} \\ & \boldsymbol{A}_{ik}\boldsymbol{\delta} + \boldsymbol{c}_k \le \boldsymbol{w} \\ & \|\boldsymbol{\delta}\|_2 \le \rho + r \\ & \boldsymbol{u} \ge \boldsymbol{0}, \boldsymbol{v} \ge \boldsymbol{0}, \boldsymbol{w} \ge \boldsymbol{0}, r \ge 0 \end{aligned} \tag{5.29a-e}$$

where $\eta > 0$ is a sufficiently large scalar [10]. With these new variables introduced, the feasible region is always nonempty. In fact, if we arbitrarily fix $\boldsymbol{\delta} = \boldsymbol{\delta}_0$ and let $\boldsymbol{u}_0 = \max\{\boldsymbol{A}_{ek}\boldsymbol{\delta}_0 + \boldsymbol{a}_k, \boldsymbol{0}\}$, $\boldsymbol{v}_0 = \max\{-(\boldsymbol{A}_{ek}\boldsymbol{\delta}_0 + \boldsymbol{a}_k), \boldsymbol{0}\}$, $\boldsymbol{w}_0 = \max\{\boldsymbol{A}_{ik}\boldsymbol{\delta}_0 + \boldsymbol{c}_k, \boldsymbol{0}\}$, and $r_0 = \max\{\|\boldsymbol{\delta}_0\|_2 - \rho, \boldsymbol{0}\}$, it can be readily verified that point $\{\boldsymbol{\delta}_0, \boldsymbol{u}_0, \boldsymbol{v}_0, \boldsymbol{w}_0, r_0\}$ is in the feasible region of the problem. The last two term in the objective function is a penalty term that, with a large $\mu > 0$, helps reduce the slack variables. Let $\{\boldsymbol{\delta}_x, \boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{w}^*, r^*\}$ be a solution of the above problem. If $\{\boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{w}^*, r^*\}$ are all zero, then vector $\boldsymbol{\delta}_x$ is taken to be a solution of the original QP problem subject to an additional bound constraint $\|\boldsymbol{\delta}\|_2 \le \rho$. If $\{\boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{w}^*, r^*\}$ are nonzero, then the constraints there must be relaxed for the problem to be solvable. Utilizing the nonzero $\{\boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{w}^*, r^*\}$, the QP subproblem in Eq. (5.23) is relaxed to

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\boldsymbol{\delta}^T \boldsymbol{Z}_k \boldsymbol{\delta} + \boldsymbol{\delta}^T \boldsymbol{g}_k \\ \text{subject to:} \quad & \boldsymbol{A}_{ek}\boldsymbol{\delta} = -\tilde{\boldsymbol{a}}_k \\ & \boldsymbol{A}_{ik}\boldsymbol{\delta} \le -\tilde{\boldsymbol{c}}_k \\ & \|\boldsymbol{\delta}\|_2 \le \tilde{\rho} \end{aligned} \qquad (5.30\text{a-d})$$

where

$$\tilde{\boldsymbol{a}}_k = \boldsymbol{a}_k - \boldsymbol{u}^* + \boldsymbol{v}^*, \;\; \tilde{\boldsymbol{c}}_k = \boldsymbol{c}_k - \boldsymbol{w}^*, \;\; \tilde{\rho} = \rho + r^* \qquad (5.31)$$

The optimal Lagrange multipliers associated with the constrains in Eqs. (5.30b) and (5.30c) are denoted by $\boldsymbol{\lambda}_{qp}$ and $\boldsymbol{\mu}_{qp}$, respectively, which are used in Eq. (5.24) to generate $\boldsymbol{\delta}_\lambda$ and $\boldsymbol{\delta}_\mu$.

5.3.4 A practical SQP algorithm for problem (5.1)

An SQP algorithm that incorporates the positive definite approximation technique in Sec. 5.3.2 and the solution technique for QP subproblem in Sec. 5.3.3 is outlined below.

Algorithm 5.6 Practical SQP algorithm for problem (5.1)

Step 1 Input initial point $\{\boldsymbol{x}_0, \boldsymbol{\lambda}_0, \boldsymbol{\mu}_0\}$, $\eta > 0$, $\tau > 0$, $\rho > 0$, $\boldsymbol{Z}_0 = \boldsymbol{I}_n$, and tolerance ε.
Set $k = 0$.

Step 2 Solve the QP problem in Eq. (5.29) to obtain $\{\boldsymbol{\delta}_x, \boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{w}^*, r^*\}$.

If $\boldsymbol{u}^* \approx \boldsymbol{0}$, $\boldsymbol{v}^* \approx \boldsymbol{0}$, $\boldsymbol{w}^* \approx \boldsymbol{0}$, and $r^* \approx 0$, then take the optimal Lagrange multiplier associated with constraints in Eqs. (5.29b) and (5.29c) as $\boldsymbol{\lambda}_{qp}$ and $\boldsymbol{\mu}_{qp}$. Otherwise, solve the QP problem in Eq. (5.30) to obtain $\{\boldsymbol{\delta}_x, \boldsymbol{\lambda}_{qp}, \boldsymbol{\mu}_{qp}\}$.

Step 3 Choose α_k using Eq. (5.27).

Step 4 Using Eq. (5.24) to obtain $\boldsymbol{\delta}_\lambda$ and $\boldsymbol{\delta}_\mu$. Set $\{\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}, \boldsymbol{\mu}_{k+1}\}$ using Eq. (5.25).

Step 5 If $\|[\boldsymbol{x}_{k+1} - \boldsymbol{x}_k;\ \boldsymbol{\lambda}_{k+1} - \boldsymbol{\lambda}_k;\ \boldsymbol{\mu}_{k+1} - \boldsymbol{\mu}_k]\|_2 < \varepsilon$, stop and output $\boldsymbol{x}_{k+1}$ as solution; otherwise update $\boldsymbol{Z}_k$ to $\boldsymbol{Z}_{k+1}$ using Eq. (5.28), set $k = k + 1$and repeat from Step 2.

Example 5.5 Apply Algorithm 5.6 to solve the constrained problem in Example 5.1, i.e.,

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) = \ln(1 + x_1^2) + x_2 \\ \text{subject to:} \quad & a(\boldsymbol{x}) = (1 + x_1^2)^2 + x_2^2 - 4 = 0 \\ & c(\boldsymbol{x}) = x_1 + x_2^2 + 0.3 \le 0 \end{aligned}$$

Solution

We follow Eq. (5.22) to compute

$$\boldsymbol{g}_k = \begin{bmatrix} \dfrac{2x_1^{(k)}}{1+x_1^{(k)2}} \\ 1 \end{bmatrix}, \quad \boldsymbol{A}_{ek} = \nabla a(\boldsymbol{x}_k)^T = \begin{bmatrix} 4x_1^{(k)}(1+x_1^{(k)2}) \\ 2x_2^{(k)} \end{bmatrix}^T, \text{ and } \boldsymbol{A}_{ik} = \nabla c(\boldsymbol{x}_k)^T = \begin{bmatrix} 1 \\ 2x_2^{(k)} \end{bmatrix}^T$$

Unlike in Example 5.1 where the initial point was taken to be $\boldsymbol{x}_0 = [-1.5\ \ 1]^T$ which violates the equality constraint $a(\boldsymbol{x}) = 0$ while satisfies the inequality constraint $c(\boldsymbol{x}) \le 0$, this time the initial point was taken to be $\boldsymbol{x}_0 = [-1.5\ \ 1.5]^T$ which violates both the equality and inequality constraints. With $\lambda_0 = 1$, $\mu_0 = 1$, $\eta = 20$, $\tau = 1$, $\rho = 1$, and $\varepsilon = 10^{-6}$, it took Algorithm 5.6 9 iterations to converge a solution $\boldsymbol{x}^* = [-0.91624199\quad -0.78501082]^T$. Fig. 5.6 shows a trajectory of the 9 iterates and a profile of the objective function versus iteration. The objective function at the solution point was found to be

$$f^* = -0.17551736.$$

Satisfaction of constraints were measured by the value of $\boldsymbol{a}(\boldsymbol{x}^*)$ which was found to be $-9.05941988 \times 10^{-14}$, and $c(\boldsymbol{x}^*) = -3.36384753 \times 10^{-10}$ which means that $c(\boldsymbol{x}^*) \le 0$ is satisfied and $c(\boldsymbol{x})$ is practically active at $\boldsymbol{x}^*$. ■

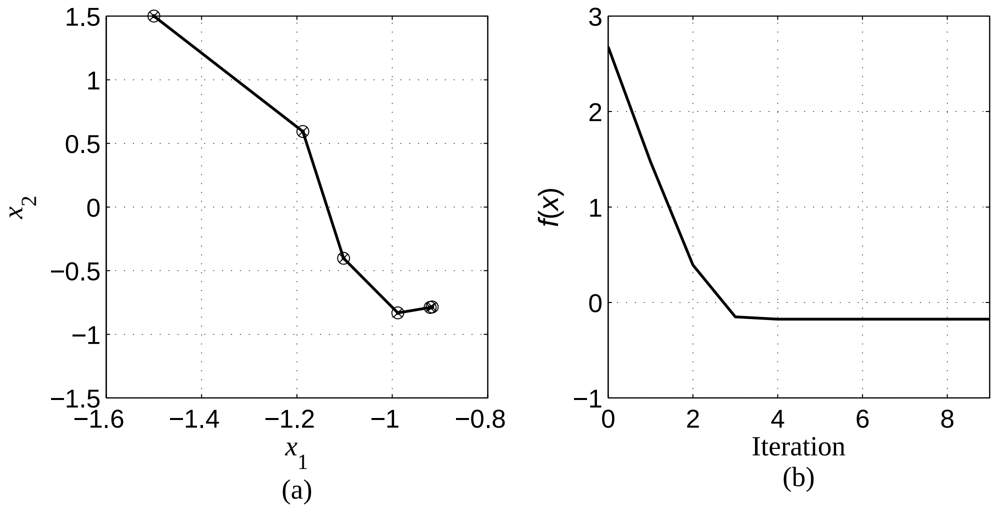


Figure 5.6. (a) First nine iterations produced by Algorithm 5.6, (b) objective function versus iteration for Example 5.4.

5.4 Convex-Concave Procedure

In this section, we explore the general optimization problem in Eq. (5.1) by using a convexification technique known as convex-concave procedure [11].

5.4.1 Basic convex-concave procedure

The convex-concave procedure (CCP) refers to a heuristic method of convexifying nonconvex problems based on the fact that a twice continuously differentiable function can be expressed as a difference of two convex functions [12]. It can also be shown that if the function in question has a bounded Hessian, such an expression can be explicitly constructed [11]. We are thus motivated to consider a nonconvex problem of the form

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = u_0(\boldsymbol{x}) - v_0(\boldsymbol{x}) \\ &\text{subject to:} \ \ c_j(\boldsymbol{x}) = u_j(\boldsymbol{x}) - v_j(\boldsymbol{x}) \le 0 \ \ \text{for } j = 1,2,...,q \end{aligned} \tag{5.32a-b}$$

where functions $u_j(\boldsymbol{x})$ and $v_j(\boldsymbol{x})$ for $j = 0, 1, \ldots, q$ are convex.

Consider the k-th iteration for the problem in Eq. (5.32), where iterate $\boldsymbol{x}_k$ is known. The CCP consists of two steps. The first step is to convexify the objective function and constraints in Eq. (5.32) by replacing each $v_j(\boldsymbol{x})$ with the affine approximation

$$\hat{v}_j(\boldsymbol{x},\boldsymbol{x}_k) = v_j(\boldsymbol{x}_k) + \nabla^T v_j(\boldsymbol{x}_k)(\boldsymbol{x}-\boldsymbol{x}_k) \quad \text{for } j = 0, 1, \ldots, q \tag{5.33}$$

As a result, the modified objective function $u_0(\boldsymbol{x}) - \hat{v}_0(\boldsymbol{x},\boldsymbol{x}_k)$ as well as constraint functions $u_j(\boldsymbol{x}) - \hat{v}_j(\boldsymbol{x},\boldsymbol{x}_k)$ are all convex. The second step is to solve the convex subproblem

$$\begin{aligned} &\text{minimize} \quad \hat{f}(\boldsymbol{x},\boldsymbol{x}_k) = u_0(\boldsymbol{x}) - \hat{v}_0(\boldsymbol{x},\boldsymbol{x}_k) \\ &\text{subject to:} \ \ \hat{c}_j(\boldsymbol{x},\boldsymbol{x}_k) = u_j(\boldsymbol{x}) - \hat{v}_j(\boldsymbol{x},\boldsymbol{x}_k) \le 0 \ \ \text{for } j = 1,2,...,q \end{aligned} \tag{5.34a-b}$$

whose solution is denoted by $\boldsymbol{x}_{k+1}$. With the new iterate $\boldsymbol{x}_{k+1}$, these two steps are repeated and a sequence of iterates $\{\boldsymbol{x}_k, k = 0,1,...\}$ is generated.

There are two issues arising from the above procedure. The first issue is whether each and every iterate in the sequence $\{\boldsymbol{x}_k, k = 0,1,...\}$ generated is feasible for the original problem in Eq. (5.32), i.e., satisfies the constraints in Eq. (5.32b). We deal with this issue by mathematical induction as follows. Suppose we start the CCP with an $\boldsymbol{x}_0$ that is feasible for the original problem, namely point $\boldsymbol{x}_0$ satisfies

$$u_j(\boldsymbol{x}_0) - v_j(\boldsymbol{x}_0) \le 0 \ \ \text{for } j = 1,2,...,q \tag{5.35}$$

From Eq. (5.33), we have $\hat{v}_j(\boldsymbol{x}_0,\boldsymbol{x}_0) = v_j(\boldsymbol{x}_0)$ which in conjunction with Eq. (5.35) gives

$$\hat{c}_j(\boldsymbol{x}_0,\boldsymbol{x}_0) = u_j(\boldsymbol{x}_0) - \hat{v}_j(\boldsymbol{x}_0,\boldsymbol{x}_0) = u_j(\boldsymbol{x}_0) - v_j(\boldsymbol{x}_0) \le 0$$

for $j = 1,2,...,q$. This means that point $\boldsymbol{x}_0$ is also a feasible point for the convex problem in Eq. (5.34), hence with $\boldsymbol{x}_0$ as an initial point the CCP can be performed. Now assume that iterate $\boldsymbol{x}_k$ is feasible for the original problem in Eq. (5.32) and point $\boldsymbol{x}_{k+1}$ is obtained by solving the convex problem in Eq. (5.34), we want to show that $\boldsymbol{x}_{k+1}$ is also feasible for the original problem. To this end, note that because $\boldsymbol{x}_{k+1}$ solves problem (5.34), it is a feasible point of the problem, hence

$$u_j(\boldsymbol{x}_{k+1}) - \hat{v}_j(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k) \le 0 \ \text{ for } j = 1, 2, ..., q \tag{5.36}$$

In addition, since function $v_j(\boldsymbol{x})$ is convex, we can write

$$v_j(\boldsymbol{x}) \ge v_j(\boldsymbol{x}_k) + \nabla^T v_j(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) = \hat{v}_j(\boldsymbol{x}, \boldsymbol{x}_k)$$

which implies that $v_j(\boldsymbol{x}_{k+1}) \ge \hat{v}_j(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k)$. This in conjunction with Eq. (5.36) leads to

$$u_j(\boldsymbol{x}_{k+1}) - v_j(\boldsymbol{x}_{k+1}) \le u_j(\boldsymbol{x}_{k+1}) - \hat{v}_j(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k) \le 0$$

for $j = 1, 2, ..., q$. Therefore $\boldsymbol{x}_{k+1}$ satisfies the constraints in Eq. (5.32b), and hence is feasible for the original problem. The second issue is whether the sequence $\{\boldsymbol{x}_k, k = 0, 1, ...\}$ generated converges. It can be shown that the objective function of the original problem, namely $u_0(\boldsymbol{x}) - v_0(\boldsymbol{x})$, decreases monotonically at $\{\boldsymbol{x}_k, k = 0, 1, ...\}$, thus CCP is a descent method [13]. To show this, let $f_k = u_0(\boldsymbol{x}_k) - v_0(\boldsymbol{x}_k)$ and note that

$$f_k = u_0(\boldsymbol{x}_k) - v_0(\boldsymbol{x}_k) = u_0(\boldsymbol{x}_k) - \hat{v}_0(\boldsymbol{x}_k, \boldsymbol{x}_k) \ge u_0(\boldsymbol{x}_{k+1}) - \hat{v}_0(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k) \tag{5.37}$$

Since $v_0(\boldsymbol{x})$ is convex, we have $v_0(\boldsymbol{x}_{k+1}) \ge \hat{v}_0(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k)$, hence

$$u_0(\boldsymbol{x}_{k+1}) - \hat{v}_0(\boldsymbol{x}_{k+1}, \boldsymbol{x}_k) \ge u_0(\boldsymbol{x}_{k+1}) - v_0(\boldsymbol{x}_{k+1}) = f_{k+1}$$

which in conjunction with Eq. (5.37) gives $f_{k+1} \le f_k$. In addition, it can be shown that the sequence $\{\boldsymbol{x}_k, k = 0, 1, \ldots\}$ converges to a critical point of the original problem in Eq. (5.32) [14]. We now summarize the algorithmic steps of the above method as

Algorithm 5.7 CCP for problem (5.32)

Step 1 Input a feasible initial point $\boldsymbol{x}_0$ and a tolerance $\varepsilon > 0$. Set $k = 0$.

Step 2 Solve the problem in Eq. (5.34). Denote the solution by $\boldsymbol{x}_{k+1}$.

Step 3 If $\left| \dfrac{f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)}{f(\boldsymbol{x}_k)} \right| < \varepsilon$, stop and output $\boldsymbol{x}_{k+1}$ as solution; otherwise set $k = k + 1$ and repeat from Step 2.

5.4.2 Penalty Convex-Concave Procedure

Algorithm 5.7 requires a point $\boldsymbol{x}_0$ that satisfies the constraints in Eq. (5.32b) to initiate the algorithm. For many practical problems that fall into the formulation in Eq. (5.32), especially those with large number of constraints, however, finding a feasible initial point can be challenging. In what follows, we describe a modified CCP, known as *penalty* CCP (PCCP) [13], that accepts an initial point $\boldsymbol{x}_0$ that needs not to be feasible for the problem in Eq. (5.32). This is made possible by introducing nonnegative slack variables $\{s_j,\ j = 1, 2, \ldots, q\}$ into the formulation that modifies the conventional CCP formulation to

$$\begin{aligned} &\text{minimize} \quad u_0(\boldsymbol{x}) - \hat{v}_0(\boldsymbol{x}, \boldsymbol{x}_k) + \tau_k \sum_{j=1}^{q} s_j \\ &\text{subject to:} \quad u_j(\boldsymbol{x}) - \hat{v}_j(\boldsymbol{x}, \boldsymbol{x}_k) \le s_j \qquad \text{for } j = 1,2,\ldots,q \\ &\qquad\qquad\quad s_j \ge 0 \qquad\qquad\qquad\qquad \text{for } j = 1,2,\ldots,q \end{aligned} \tag{5.38a-c}$$

To explain the role the slack variables play, consider an initial point $\boldsymbol{x}_0$ which violates only one of the constraints in Eq. (5.32b), say the j^*-th constraint, i.e.,

$$z_{j^*} \triangleq u_{j^*}(\boldsymbol{x}_0) - v_{j^*}(\boldsymbol{x}_0) = u_{j^*}(\boldsymbol{x}_0) - \hat{v}_{j^*}(\boldsymbol{x}_0, \boldsymbol{x}_0) > 0$$

For the modified formulation in Eq. (5.38), point $\boldsymbol{x}_0$ is acceptable because Eqs. (5.38a-c) involve both $\boldsymbol{x}$ and $\boldsymbol{s} = \begin{bmatrix} s_1 & s_2 & \cdots & s_q \end{bmatrix}^T$ as the design variables, and an initial point $\{\boldsymbol{x}_0, \boldsymbol{s}_0\}$ is obviously feasible because it satisfies the constraints in Eq. (5.38b) as long as $\boldsymbol{s}_0 \ge \boldsymbol{0}$ and its j^*-th component is no less than z_{j^*}. Once the algorithm gets started with an easy-to-find feasible initial point, the values of $\{s_j, j = 1, 2, \ldots, q\}$ have to be reduced, gradually approaching to zero, so that the iterates obtained by solving the convex problem in Eq. (5.38) approaches to the solution of the problem in Eq. (5.34). The gradual reduction of slack variables s_j is achieved by including a penalty term, $\tau_k \sum_{j=1}^{q} s_j$, in the objective function in Eq. (5.38a) where $\tau_k > 0$ is a scalar weight that controls the trade-off between the original objective function and the penalty term. The algorithm starts with a reasonable value τ_0, say $\tau_0 = 1$, and the value of τ monotonically increases as the algorithm proceeds until it reaches an upper limit $\tau_{\max}$. In this way, the penalty term in Eq. (5.38a) forces a quick reduction of $\{s_j, j = 1, 2, \ldots, q\}$. As soon as all s_j's are nullified, they remain to be zero because of the presence of the penalty term in the objective function. From this point on, the problem in Eq. (5.38) is practically identical to that in Eq. (5.34). The upper limit $\tau_{\max}$ is imposed to avoid numerical difficulties that may occur if $\tau_{\max}$ becomes too large [13]. Below we summarize the above analysis as a PCCP-based algorithm for the problem in Eq. (5.32).

Algorithm 5.8 Penalty convex-concave procedure for problem (5.32)

Step 1 Input an initial point $\boldsymbol{x}_0$, $\tau_0 = 1$, $\tau_{\max} > 0$, $\mu > 1$, and a tolerance $\varepsilon > 0$. Set $k = 0$.

Step 2 Generate a feasible initial point $\{\boldsymbol{x}_0, \boldsymbol{s}_0\}$ where $\boldsymbol{s}_0 = \begin{bmatrix} s_1^{(0)} & s_2^{(0)} & \cdots & s_q^{(0)} \end{bmatrix}^T$ with

$s_j^{(0)} = \max\{0.1, u_j(\boldsymbol{x}_0) - v_j(\boldsymbol{x}_0) + 0.1\}$ for $j = 1, 2, \ldots, q$.

Step 3 Solve the problem in Eq. (5.38). Denote the solution by $\boldsymbol{x}_{k+1}$.

Step 4 If $\left| \dfrac{f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k)}{f(\boldsymbol{x}_k)} \right| < \varepsilon$, stop and output $\boldsymbol{x}_{k+1}$ as solution; otherwise set

$\tau_{k+1} = \min\{\mu\tau_k, \tau_{\max}\}$, $k = k + 1$ and repeat from Step 2.

Example 5.6 Apply Algorithm 5.8 to solve the constrained problem in Example 5.1, i.e.,

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) = \ln(1 + x_1^2) + x_2 \\ \text{subject to:} \quad & a(\boldsymbol{x}) = (1 + x_1^2)^2 + x_2^2 - 4 = 0 \\ & c(\boldsymbol{x}) = x_1 + x_2^2 + 0.3 \le 0 \end{aligned}$$

Solution

First, we re-formulate the problem at hand as

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) = \ln(1 + x_1^2) + x_2 \\ \text{subject to:} \quad & c_1(\boldsymbol{x}) = 4 - (1 + x_1^2)^2 - x_2^2 \le 0 \\ & c_2(\boldsymbol{x}) = (1 + x_1^2)^2 + x_2^2 - 4 \le 0 \\ & c_3(\boldsymbol{x}) = x_1 + x_2^2 + 0.3 \le 0 \end{aligned} \tag{5.39a-d}$$

to fit into the form in Eq. (5.32). Note that the objective function $f(\boldsymbol{x})$ and constraint function $c_1(\boldsymbol{x})$ are not convex. The objective function can be expressed as $f(\boldsymbol{x}) = u_0(\boldsymbol{x}) - v_0(\boldsymbol{x})$ where $u_0(\boldsymbol{x}) = \ln(1 + x_1^2) + \frac{1}{2}x_1^2 + x_2$ and $v_0(\boldsymbol{x}) = \frac{1}{2}x_1^2$ are convex because the Hessian

$$\nabla^2 u_0(\boldsymbol{x}) = \text{diag}\{\tfrac{x_1^4+3}{(x_1^2+1)^2}, 0\} \text{ and } \nabla^2 v_0(\boldsymbol{x}) = \text{diag}\{1, 0\}$$

are positive semidefinite. The constraint function $c_1(\boldsymbol{x})$ is in the form $c_1(\boldsymbol{x}) = u_1(\boldsymbol{x}) - v_1(\boldsymbol{x}) \le 0$ with $u_1(\boldsymbol{x}) = 4$ and $v_1(\boldsymbol{x}) = (1 + x_1^2)^2 + x_2^2$ both of which are obviously convex. In the k-th iteration of the algorithm, affine approximations of $v_0(\boldsymbol{x})$ and $v_1(\boldsymbol{x})$ at $\boldsymbol{x}_k$ are given by

$$\hat{v}_0(\boldsymbol{x}, \boldsymbol{x}_k) = \tfrac{1}{2}(x_1^{(k)})^2 + [x_1^{(k)} \quad 0](\boldsymbol{x} - \boldsymbol{x}_k)$$

and

$$\hat{v}_1(\boldsymbol{x}, \boldsymbol{x}_k) = v_1(\boldsymbol{x}_k) + [4x_1^{(k)}(1 + (x_1^{(k)})^2) \quad 2x_2^{(k)}](\boldsymbol{x} - \boldsymbol{x}_k)$$

respectively. Because of the convexity of $c_2(\boldsymbol{x})$ and $c_3(\boldsymbol{x})$, the last two constraints do not require convex approximations and we simply take $u_2(\boldsymbol{x}) = c_2(\boldsymbol{x})$, $v_2(\boldsymbol{x}) = 0$, $u_3(\boldsymbol{x}) = c_3(\boldsymbol{x})$, and $v_3(\boldsymbol{x}) = 0$. Consequently, the convex problem to be solved in the kth iteration assumes the form

$$\begin{aligned}
\text{minimize} \quad & \ln(1 + x_1^2) + \tfrac{1}{2}x_1^2 + x_2 - \tfrac{1}{2}(x_1^{(k)})^2 - [x_1^{(k)} \quad 0](\boldsymbol{x} - \boldsymbol{x}_k) + \tau_k \textstyle\sum_{j=1}^{3} s_j \\
\text{subject to:} \quad & 4 - v_1(\boldsymbol{x}_k) - [4x_1^{(k)}(1 + (x_1^{(k)})^2) \quad 2x_2^{(k)}](\boldsymbol{x} - \boldsymbol{x}_k) \le s_1 \\
& (1 + x_1^2)^2 + x_2^2 - 4 \le s_2 \\
& x_1 + x_2^2 + 0.3 \le s_3 \\
& s_j \ge 0 \ \text{ for } j = 1, 2, 3.
\end{aligned} \tag{5.40}$$

To examine how the algorithm handles an initial $\boldsymbol{x}_0$ that violates some of the constraints in the problem in Eq. (5.39), we choose $\boldsymbol{x}_0 = [-1.5 \quad 1.2]^T$ which satisfies the constraint in Eq. (5.39b) but violates the constraints in Eq. (5.32c) and (5.32d). With this $\boldsymbol{x}_0$ given, Step 2 of the algorithm yields an initial slack variable $\boldsymbol{s}_0 = [0.1 \ 7.8725 \ 0.11]^T$ which assures the feasibility of the initial point $\{\boldsymbol{x}_0, \boldsymbol{s}_0\}$which is feasible for the convex problem in Eq. (5.40).

With $\tau_0 = 1, \tau_{\max} = 10^3, \mu = 1.5$, and $\varepsilon = 10^{-8}$, it took Algorithm 5.8 14 iterations to converge to a solution $\boldsymbol{x}^* = [-0.91624199 \quad -0.78501082]^T$. The objective function at the solution point was found to be $f^* = -0.17551735$. Satisfaction of constraints were measured by the value of $\boldsymbol{a}(\boldsymbol{x}^*)$ which was found to be $-1.77545978\times10^{-10}$, and $c(\boldsymbol{x}^*) = -6.92222374\times10^{-9}$ which means that $c(\boldsymbol{x}^*) \le 0$ is satisfied and $c(\boldsymbol{x})$ is practically active at $\boldsymbol{x}^*$. Fig. 5.7 shows a trajectory of the fourteen iterates and a profile of the objective function versus PCCP iterations. ■

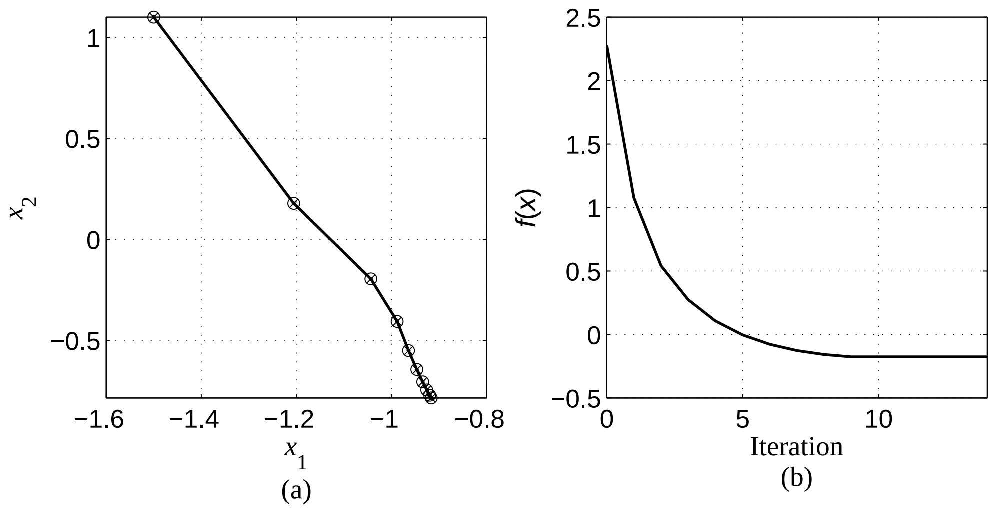


Figure 5.7. (a) First fourteen iterations produced by Algorithm 5.8, (b) objective function versus iteration for Example 5.5.

5.5 ADMM heuristic for nonconvex problems

The alternating direction methods of multipliers (ADMM) for convex problems are studied in Sec. 4.5. In this section, ADMM is extended as a heuristic to some nonconvex problems. We consider the class of constrained problems [15, Sec. 9.1] which assume the form

$$\begin{aligned} &\text{minimize} \;\; f(\boldsymbol{x}) \\ &\text{subject to:} \;\; \boldsymbol{x} \in C \end{aligned} \qquad \text{(5.41a-b)}$$

where function $f(\boldsymbol{x})$ is convex but the feasible region C is a nonconvex, hence Eq. (5.41) formulates a class of nonconvex problems. On comparing the formulation in Eq. (5.41) with that in Eq. (4.94), the two problem formulations look quite similar except the convexity of the feasible region involved: the set C in Eq. (4.94) is convex while the set C in Eq. (5.41) is not. It is therefore intuitively reasonable that an ADMM heuristic approach be developed by extending the techniques used for the problem in Eq. (4.94) to the problem in Eq. (5.41). First, the problem in Eq. (5.41) is re-formulated as

$$\text{minimize} \quad f(\boldsymbol{x}) + I_C(\boldsymbol{x})$$

where $I_C(\boldsymbol{x})$ is the indicator function associated with set C. Next, new variable $\boldsymbol{y}$ is introduced to split the objective in the above problem and, as a result, the problem at hand is converted to

$$\begin{aligned} &\text{minimize} \;\; f(\boldsymbol{x}) + I_C(\boldsymbol{y}) \\ &\text{subject to:} \;\; \boldsymbol{x} - \boldsymbol{y} = \boldsymbol{0} \end{aligned} \qquad \text{(5.42)}$$

Similar to the scaled ADMM iterations in Sec. 14.5.2, Eq. (5.42) suggests the scaled ADMM iterations

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \tfrac{\alpha}{2} \| \boldsymbol{x} - \boldsymbol{y}_k + \boldsymbol{v}_k \|_2^2 \right] \\ \boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}} \left[I_C(\boldsymbol{y}) + \tfrac{\alpha}{2} \| \boldsymbol{y} - (\boldsymbol{x}_{k+1} + \boldsymbol{v}_k) \|_2^2 \right] \\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1} \end{aligned}$$

where the $\boldsymbol{x}$-minimization is obviously a convex problem because $f(\boldsymbol{x})$ is convex while the y-minimization can be obtained by minimizing $\| \boldsymbol{y} - (\boldsymbol{x}_{k+1} + \boldsymbol{v}_k) \|_2$ subject to $\boldsymbol{y} \in C$. This means that $\boldsymbol{y}_{k+1}$ can be computed by projecting $\boldsymbol{x}_{k+1} + \boldsymbol{v}_k$ onto set C, and hence the ADMM iterations can be expressed as

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \tfrac{\alpha}{2} \| \boldsymbol{x} - \boldsymbol{y}_k + \boldsymbol{v}_k \|_2^2 \right] \\ \boldsymbol{y}_{k+1} &= P_C(\boldsymbol{x}_{k+1} + \boldsymbol{v}_k) \\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1} \end{aligned} \qquad \text{(5.43a-c)}$$

where $P_C(\boldsymbol{z})$ denote the projection of point $\boldsymbol{z}$ onto *nonconvex* set C. It is the projection in Eq.

(5.43b) that differs from that in Eq. (4.97b) and is difficult to calculate in general as it involves a nonconvex feasible region $\mathcal{C}$. As demonstrated in [15, Sec. 9.1], however, there are several important cases where the projection in Eq. (5.43b) can be carried out precisely. Based on the analysis, an ADMM-based algorithm for the nonconvex problem in Eq. (5.41) can be outlined as follows.

Algorithm 5.9 Scaled ADMM for Problem (5.34)

Step 1 Input parameter $\alpha > 0$, $\boldsymbol{y}_0$, $\boldsymbol{v}_0$, and tolerance $\varepsilon_p > 0$, $\varepsilon_d > 0$.

Set $k = 0$.

Step 2 Compute

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}}\left[f(\boldsymbol{x}) + \tfrac{\alpha}{2}\|\boldsymbol{x} - \boldsymbol{y}_k + \boldsymbol{v}_k\|_2^2\right]\\ \boldsymbol{y}_{k+1} &= P_{\mathcal{C}}(\boldsymbol{x}_{k+1} + \boldsymbol{v}_k)\\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}\end{aligned}$$

Step 3 Compute dual residual

$$\boldsymbol{d}_k = -\alpha(\boldsymbol{y}_k - \boldsymbol{y}_{k-1})$$

and primal residual

$$\boldsymbol{r}_k = \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}$$

Step 4 If

$$\|\boldsymbol{r}_k\|_2 < \varepsilon_p \quad \text{and} \quad \|\boldsymbol{d}_k\|_2 < \varepsilon_d$$

output ($\boldsymbol{x}_{k+1}$, $\boldsymbol{y}_{k+1}$) as solution and stop; Otherwise, set $k = k + 1$ and repeat from Step 2.

Example 5.7 Apply Algorithm 5.9 to solve the constrained problem

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x}) = x_2^2 - 2x_1 + x_2\\ &\text{subject to:} \quad x_1^2 + x_2^2 - 9 = 0\end{aligned}$$

Solution

We start by writing the objective function involved in Eq. (5.43a) as

$$\begin{aligned}&x_2^2 - 2x_1 + x_2 + \tfrac{\alpha}{2}\|\boldsymbol{x} - \boldsymbol{y}_k + \boldsymbol{v}_k\|_2^2\\ &= \boldsymbol{x}^T\begin{bmatrix}\alpha/2 & 0\\ 0 & 1+\alpha/2\end{bmatrix}\boldsymbol{x} - \boldsymbol{x}^T\left(\alpha(\boldsymbol{y}_k - \boldsymbol{v}_k) - \begin{bmatrix}2\\ -1\end{bmatrix}\right)\end{aligned}$$

By minimizing the above objective function, we obtain

$$\boldsymbol{x}_{k+1} = \begin{bmatrix}1/\alpha & 0\\ 0 & 1/(2+\alpha)\end{bmatrix}\left(\alpha(\boldsymbol{y}_k - \boldsymbol{v}_k) - \begin{bmatrix}2\\ -1\end{bmatrix}\right)$$

The feasible region $\mathcal{C}$ is a circle with center at the origin and radius 3, namely,

$$C = \{\boldsymbol{x} : x_1^2 + x_2^2 = 9\} \tag{5.44}$$

The step in Eq. (5.43b) is carried out by projecting point $\boldsymbol{x}_{k+1} + \boldsymbol{v}_k$ onto the circle C defined by Eq. (5.44). If we let the two coordinates of $\boldsymbol{x}_{k+1} + \boldsymbol{v}_k$ be p_1 and p_2, respectively, and the two coordinates of the projection $P_C(\boldsymbol{x}_{k+1} + \boldsymbol{v}_k)$ be q_1 and q_2, then it can readily be verified that (i) if $p_1 = 0$ and $p_2 > 0$, then $q_1 = 0$ and $q_2 = 3$; (ii) if $p_1 = 0$ and $p_2 < 0$, then $q_1 = 0$ and $q_2 = -3$; (iii) if $p_1 > 0$, then $q_1 = t$ and $q_2 = t \cdot p_2/p_1$; and (iv) if $p_1 < 0$, then $q_1 = -t$ and $q_2 = -t \cdot p_2/p_1$, where $t = 3/\sqrt{1+(p2/p1)^2}$.

With $\alpha = 0.8$, $\varepsilon_p = 10^{-4}$, and $\varepsilon_d = 10^{-4}$, it took Algorithm 5.9 12 iterations to converge to a vector $\hat{\boldsymbol{x}}^*$ which was projected onto the feasible region C to yield the solution

$$\boldsymbol{x}^* = [2.97656348 \quad -0.37425907]^T$$

The 2-norm of the primal residual $\boldsymbol{r}_k$ and dual residual $\boldsymbol{d}_k$ versus iteration are shown in Fig. 5.8. ■

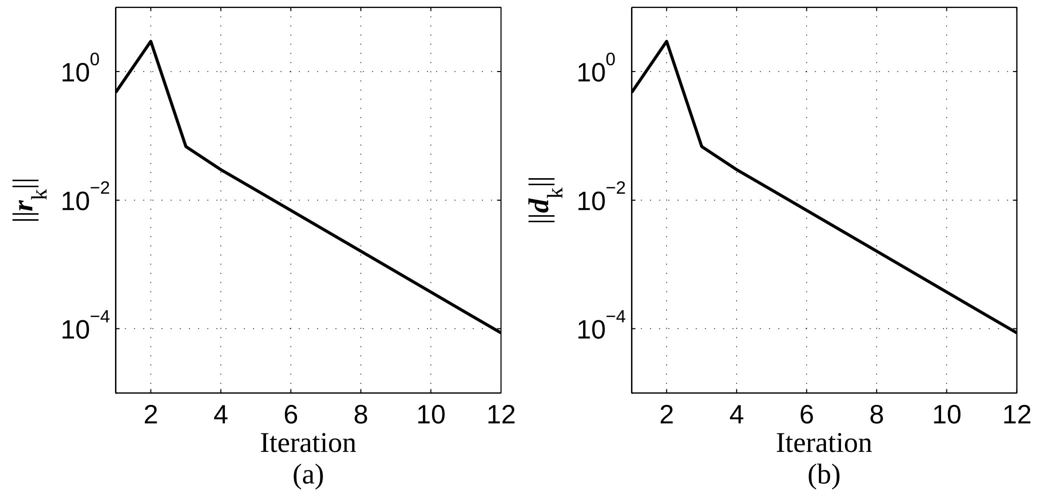


Figure 5.8. (a) 2-norm of primal residual $\| \boldsymbol{r}_k \|_2$, (b) 2-norm of dual residual $\| \boldsymbol{d}_k \|_2$ for Example 5.6.

References

[1] S. Boyd, Course Notes for EE364b: *Convex Optimization II*, Stanford University, available online: http://stanford.edu/class/ee364b/lectures.html.

[2] R. B. Wilson, *A Simplicial Algorithm for Concave Programming*, Ph.D. dissertation, Graduate School of Business Administration, Harvard University, Cambridge, MA., 1963.

[3] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, pp. 1-52, 1995.

[4] S. P. Han, “A globally convergent method for nonlinear programming,” *J. Optimization Theory and Applications*, vol. 22, pp. 297-309, July 1977.

[5] R. H. Byrd and J. Nocedal, “An analysis of reduced Hessian methods for constrained optimization,” *Math. Programming*, vol. 49, pp. 285-323, 1991.

[6] R. Fletcher, “A class of methods for nonlinear programming, iii: Rates of convergence,” in *Numerical Methods for Nonlinear Optimization*, F. A. Lootsma ed., Academic Press, New York, 1972.

[7] M. J. D. Powell and Y. Yuan, “A recursive quadratic programming algorithm that uses differentiable exact penalty functions,” *Math. Programming*, vol. 35, pp. 265-278, 1986.

[8] P. T. Boggs and J. W. Tolle, “A strategy for global convergence in sequential quadratic programming algorithm,” *SIAM J. Numerical Analysis*, vol. 21, pp. 600-623, 1989.

[9] M. J. D. Powell, “A fast algorithm for nonlinearly constrained optimization calculations,” *Numerical Analysis*, vol. 630 of the series *Lecture Notes in Mathematics*, pp. 144-157, 1978.

[10] J. N. Nocedal and S. J. Wright, Numerical Optimization, 2nd edition, Springer, New York, 2006.

[11] A. L. Yuille and A. Rangarajan, “The concave-convex procedure,” *Neural Computation*, vol. 15, no. 4, pp. 915–936, 2003.

[12] P. Hartman, “On functions representable as a difference of convex functions,” *Pacific J. of Math.*, vol. 9, no. 3, pp. 707–713, 1959.

[13] T. Lipp and S. Boyd, “Variations and extensions of the convex-concave procedure,” *Optimization and Engineering*, vol. 17, no. 2, pp. 263-287, 2016.

[14] G. R. Lanckreit and B. K. Sriperumbudur, “On the convergence of the concave-convex procedure,” in *Advances in Neural Information Processing Systems*, pp. 1759–1767, 2009.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2010.

[16] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, pp. 30–37, August 2009.

Problems

5.1 Apply Algorithm 5.3 to the problem

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x}) = \ln(1+x_1^2) + x_2\\&\text{subject to:} \quad a(\boldsymbol{x}) = (1+x_1^2)^2 + x_2^2 - 4 = 0\end{aligned}$$

Start with $\boldsymbol{x}_0 = \begin{bmatrix}1 & 1\end{bmatrix}^T$ and examine the solution obtained.

5.2 Apply Algorithm 5.6 to the problem in Prob. 5.1 with the same initial point. Examine the solution obtained.

5.3 Apply Algorithm 5.3 to the problem

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x}) = x_1^4 + 2x_2^4 + x_3^4 - x_1^2x_2^2 - x_1^2x_3^2\\&\text{subject to:} \quad a_1(\boldsymbol{x}) = x_1^4 + x_2^4 + x_3^4 - 25 = 0\\&\qquad\qquad\quad a_2(\boldsymbol{x}) = 8x_1^2 + 14x_2^2 + 7x_3^2 - 56 = 0\end{aligned}$$

Start with $\boldsymbol{x}_0 = \begin{bmatrix}3 & 2 & 3\end{bmatrix}^T$ and examine the solution obtained.

5.4 Apply Algorithm 5.6 to the problem in Prob. 5.3 with the same initial point. Examine the solution obtained.

5.5 Apply Algorithm 5.3 to the problem [10, Chap. 18]

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x}) = e^{x_1x_2x_3x_4x_5} - \tfrac{1}{2}(x_1^3 + x_2^3 + 1)^2\\&\text{subject to:} \quad a_1(\boldsymbol{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 11 = 0\\&\qquad\qquad\quad a_2(\boldsymbol{x}) = x_2x_3 - 5x_4x_5 = 0\\&\qquad\qquad\quad a_3(\boldsymbol{x}) = x_1^3 + x_2^3 + 1 = 0\end{aligned}$$

Start with $\boldsymbol{x}_0 = \begin{bmatrix}-1.71 & 1.59 & 1.82 & -0.763 & -0.763\end{bmatrix}^T$ and examine the solution obtained.

5.6 Apply the alternating minimization algorithm to solve the problem in Example 5.4 with a slightly reduced data set:

Viewer\Movie	Doctor Strange	Star Trek: Beyond	Zootopia
Alice	1	?	5
Dave	3	4	?
Joe	?	5	2

where Joe's rating on Doctor Strange is unknown. Use the same initial point as in Example 5.4 and try $\mu = 10/3$, $\lambda = 0.0001$, and 400 rounds of iterations. Compare your results with those obtained in Example 5.4.

5.7 Solve the problem in Example 5.4 by using a simplified gradient descent (GD) algorithm where step size α_k is fixed to a small constant, i.e. $\alpha_k \equiv \alpha$ for all *k*. As in Example 5.4, use the same dimension d = 3 for latent space. Throughout the algorithm, variable $\boldsymbol{x}$ includes all unknown parameters, and is a column vector of dimension 24 as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \\ \boldsymbol{u}_3 \\ \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \boldsymbol{v}_3 \\ b_1^{(1)} \\ b_2^{(1)} \\ b_3^{(1)} \\ b_1^{(2)} \\ b_2^{(2)} \\ b_3^{(2)} \end{bmatrix}$$

Follow the steps below to carry out the optimization procedure.

(a) Derive a closed-form expression for the gradient of the objective function $f(\boldsymbol{x})$ which is defined by Eq. (5.14).

(b) Update iterate $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k)$, where $\alpha > 0$ is a constant.

(c) With $\mu = 23/7, \lambda = 0.01,\ \alpha = 0.1$, and a randomly selected initial point $\boldsymbol{x}_0$

```
randn('state',17)
x0 = 0.01*randn(24,1);
```

perform 400 GD iterations. Compare your results with those obtained in Example 5.4.

5.8 (a) Apply alternating minimization (AM) (see Example 5.4) to solve the following 5-star rating problem involving six users and six products:

User\Product	#1	#2	#3	#4	#5	#6
#1	3	2	5	?	3	4
#2	?	4	?	4	4	5
#3	2	4	4	?	?	5
#4	3	3	3	5	5	4
#5	4	4	5	3	4	?
#6	2	?	4	4	5	3

(b) Solve the problem in part (a) using the GD method utilized in Prob. 5.7.

5.9 Apply Algorithm 5.6 to the problem in Prob. 5.5 with the same initial point. Examine the solution obtained.

5.10 Apply Algorithm 5.7 to the constrained problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = -2x_1x_2 \\ &\text{subject to:} \ c_1(\boldsymbol{x}) = 0.2x_1^2 + 0.4x_2^2 - 1 \le 0 \end{aligned}$$

(a) Start with $\boldsymbol{x}_0 = [0 \quad 0]^T$ and examine the solution obtained.

(b) Start with $\boldsymbol{x}_0 = [-2 \quad -2]^T$ and examine the solution obtained.

5.11 Apply Algorithm 5.8 to the constrained problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = (x_1 - 1)^2 + (x_2 - 0.5)^2 \\ &\text{subject to:} \ c_1(\boldsymbol{x}) = 2x_1^2 - x_2 \le 0 \\ &\qquad\qquad\quad c_2(\boldsymbol{x}) = x_1 - 2x_2^2 \le 0 \end{aligned}$$

Start with $\boldsymbol{x}_0 = [0.2 \quad -0.2]^T$ and examine the solution obtained.

5.12 Apply Algorithm 5.8 to the constrained problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = x_1^2 + 4x_1x_2 + 3x_2^2 \\ &\text{subject to:} \ c_1(\boldsymbol{x}) = -x_1^2 + x_2 \le 0 \\ &\qquad\qquad\quad c_2(\boldsymbol{x}) = x_1^2 + x_2^2 - 1 \le 0 \end{aligned}$$

Start with $\boldsymbol{x}_0 = [-0.5 \quad 4]^T$ and examine the solution obtained.

5.13 Apply Algorithm 5.8 to the constrained problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1 \\ &\text{subject to:} \ c_1(\boldsymbol{x}) = -x_1 - 3 \le 0 \\ &\qquad\qquad\quad c_2(\boldsymbol{x}) = x_1 - 4 \le 0 \\ &\qquad\qquad\quad c_3(\boldsymbol{x}) = -x_2 - 3 \le 0 \\ &\qquad\qquad\quad c_4(\boldsymbol{x}) = x_2 - 4 \le 0 \end{aligned}$$

Start with $\boldsymbol{x}_0 = [-3.1 \quad 4.1]^T$ and examine the solution obtained.

5.14 Verify that the KKT conditions of the QP problem in Eq. (5.23) are given by Eq. (5.21).

5.15 Apply Algorithm 5.6 to the problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = 0.01x_1^2 + x_2^2 \\ &\text{subject to:} \ c_1(\boldsymbol{x}) = x_1x_2 - 25 \le 0 \\ &\qquad\qquad\quad c_2(\boldsymbol{x}) = -x_1^2 - x_2^2 + 25 \le 0 \\ &\qquad\qquad\quad c_3(\boldsymbol{x}) = -x_1 + 2 \le 0 \end{aligned}$$

Start with $\boldsymbol{x}_0 = [15 \quad 5]^T$ and examine the solution obtained.

5.16 Apply Algorithm 5.8 to the problem in Prob. 5.15 with the same initial point. Examine the solution obtained.

5.17 Apply Algorithm 5.6 to the constrained problem

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x}) = 2x_1 + x_2^2\\&\text{subject to:} \quad a_1(\boldsymbol{x}) = 4x_1^2 + x_2^2 - 9 = 0\\&\qquad\qquad\quad \boldsymbol{Ax} \le \boldsymbol{b}\end{aligned}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 0 & -1 \\ 0 & 1 \\ -2 & 0 \\ 2 & 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{bmatrix} -1 \\ 5 \\ -2 \\ 4 \end{bmatrix}$$

Start with $\boldsymbol{x}_0 = [6 \quad -2]^T$ and examine the solution obtained.

5.18 Apply Algorithm 5.8 to the problem in Prob. 5.17 with the same initial point. Examine the solution obtained.

5.19 Apply Algorithm 5.6 to the constrained problem

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x}) = 2x_1^2 + 3x_2^2\\&\text{subject to:} \quad a_1(\boldsymbol{x}) = x_1^2 + x_2^2 - 16 = 0\\&\qquad\qquad\quad \boldsymbol{Ax} \le \boldsymbol{b}\end{aligned}$$

where

$$\boldsymbol{A} = \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad \boldsymbol{b} = \begin{bmatrix} -2 \\ 5 \\ -1 \\ 5 \end{bmatrix}$$

Start with $\boldsymbol{x}_0 = [-1 \quad -2]^T$ and examine the solution obtained.

5.20 Apply Algorithm 5.8 to the problem in Prob. 5.19 with the same initial point. Examine the solution obtained.

5.21 Apply Algorithm 5.9 to solve the least-squares problem with binary variables

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x}) = \tfrac{1}{2}\|\boldsymbol{Ax} - \boldsymbol{b}\|_2^2\\&\text{subject to:} \quad x_i \in \{1, -1\} \ \text{ for } \ i = 1, 2, \ldots, n\end{aligned}$$

where $\{\boldsymbol{A} \in R^{m\times n}, \boldsymbol{b} \in R^{m\times 1}\}$ with m = 50 and n = 12 are randomly generated as follows:

```
randn('state',17)
A = randn(50,12);
randn('state',6)
b = 5*randn(50,1);
```

Appendix A

Basic Principles and Techniques of Linear Algebra

A.1 Introduction

In this appendix we summarize some basic principles of linear algebra [1]–[4] that are needed to understand the derivation and analysis of the optimization algorithms and techniques presented in the book. We state these principles without derivations. However, a reader with an undergraduate-level linear-algebra background should be in a position to deduce most of them without much difficulty. Indeed, we encourage the reader to do so as the exercise will contribute to the understanding of the optimization methods described in this book.

In what follows, R^n denotes a vector space that consists of all column vectors with n real-valued components, and C^n denotes a vector space that consists of all column vectors with n complex-valued components. Likewise, $R^{m\times n}$ and $C^{m\times n}$ denote spaces consisting of all $m \times n$ matrices with real-valued and complex-valued components, respectively. Evidently, $R^{m\times 1} \equiv R^m$ and $C^{m\times 1} \equiv C^m$. Boldfaced uppercase letters, e.g., $\mathbf{A},\ \mathbf{M}$, represent matrices, and boldfaced lowercase letters, e.g., $\mathbf{a},\ \mathbf{x}$, represent column vectors. $\mathbf{A}^T$ and $\mathbf{A}^H = (\mathbf{A}^*)^T$ denote the transpose and complex-conjugate transpose of matrix $\mathbf{A}$, respectively. $\mathbf{A}^{-1}$ (if it exists) and det$(\mathbf{A})$ denote the inverse and determinant of square matrix $\mathbf{A}$, respectively. The identity matrix of dimension n is denoted as $\mathbf{I}_n$. Column vectors will be referred to simply as vectors henceforth for the sake of brevity.

A.2 Linear Independence and Basis of a Span

A number of vectors $\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_k$ in R^n are said to be *linearly independent* if

$$\sum_{i=1}^{k} \alpha_i \mathbf{v}_i = \mathbf{0} \tag{A.1}$$

only if $\alpha_i = 0$ for $i = 1,\ 2,\ \ldots,\ k$. Vectors $\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_k$ are said to be *linearly dependent* if there exit real scalars α_i for $i = 1,\ 2,\ \ldots,\ k$, with at least one nonzero α_i, such that Eq. (A.1) holds.

A subspace $\mathcal{S}$ is a subset of R^n such that $\mathbf{x} \in \mathcal{S}$ and $\mathbf{y} \in \mathcal{S}$ imply that $\alpha\mathbf{x} + \beta\mathbf{y} \in \mathcal{S}$ for any real scalars α and β. The set of all linear combinations of vectors $\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_k$ is a subspace called the *span* of $\{\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_k\}$ and is denoted as span$\{\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_k\}$.

Given a set of vectors $\{\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_k\}$, a subset of r vectors $\{\mathbf{v}_{i_1},\ \mathbf{v}_{i_2},\ \ldots,\ \mathbf{v}_{i_r}\}$ is said to be a maximal linearly independent subset if (a) vectors $\mathbf{v}_{i_1},\ \mathbf{v}_{i_2},\ \ldots,\ \mathbf{v}_{i_r}$ are linearly independent, and (b) any vector in $\{\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_k\}$ can be expressed as a linear combination of $\mathbf{v}_{i_1},\ \mathbf{v}_{i_2},\ \ldots,\ \mathbf{v}_{i_r}$. In such a case, the vector set $\{\mathbf{v}_{i_1},\ \mathbf{v}_{i_2},\ \ldots,\ \mathbf{v}_{i_r}\}$ is called a *basis* for span$\{\mathbf{v}_1,\ \mathbf{v}_2,\ \ldots,\ \mathbf{v}_k\}$ and integer r is called the *dimension* of the subspace The dimension of a subspace $\mathcal{S}$ is denoted as dim$(\mathcal{S})$.

Example A.1 Examine the linear dependence of vectors

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ -1 \\ 3 \\ 0 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 0 \\ 2 \\ 1 \\ -1 \end{bmatrix}, \quad \mathbf{v}_3 = \begin{bmatrix} 3 \\ -7 \\ 7 \\ 2 \end{bmatrix}, \quad \text{and} \quad \mathbf{v}_4 = \begin{bmatrix} -1 \\ 5 \\ -1 \\ -2 \end{bmatrix}$$

and obtain a basis for span$\{\mathbf{v}_1,\ \mathbf{v}_2,\ \mathbf{v}_3,\ \mathbf{v}_4\}$.

Solution We note that

$$3\mathbf{v}_1 + 2\mathbf{v}_2 - 2\mathbf{v}_3 - 3\mathbf{v}_4 = \mathbf{0} \tag{A.2}$$

Hence vectors $\mathbf{v}_1,\ \mathbf{v}_2,\ \mathbf{v}_3$, and $\mathbf{v}_4$ are linearly dependent. If

$$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 = \mathbf{0}$$

then

$$\begin{bmatrix} \alpha_1 \\ -\alpha_1 + 2\alpha_2 \\ 3\alpha_1 \\ -\alpha_2 \end{bmatrix} = \mathbf{0}$$

which implies that $\alpha_1 = 0$ and $\alpha_2 = 0$. Hence $\mathbf{v}_1$ and $\mathbf{v}_2$ are linearly independent. We note that

$$\mathbf{v}_3 = 3\mathbf{v}_1 - 2\mathbf{v}_2 \tag{A.3}$$

and by substituting Eq. (A.3) into Eq. (A.2), we obtain

$$-3\mathbf{v}_1 + 6\mathbf{v}_2 - 3\mathbf{v}_4 = \mathbf{0}$$

i.e.,

$$\mathbf{v}_4 = -\mathbf{v}_1 + 2\mathbf{v}_2 \tag{A.4}$$

Thus vectors $\mathbf{v}_3$ and $\mathbf{v}_4$ can be expressed as linear combinations of $\mathbf{v}_1$ and $\mathbf{v}_2$. Therefore, $\{\mathbf{v}_1,\ \mathbf{v}_2\}$ is a basis of span$\{\mathbf{v}_1,\ \mathbf{v}_2,\ \mathbf{v}_3,\ \mathbf{v}_4\}$.

■

A.3 Range, Null Space, and Rank

Consider a system of linear equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{A.5}$$

where $\mathbf{A} \in R^{m\times n}$ and $\mathbf{b} \in R^{m\times 1}$. If we denote the ith column of matrix $\mathbf{A}$ as $\mathbf{a}_i \in R^{m\times 1}$, i.e.,

$$\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots\ \mathbf{a}_n]$$

and let

$$\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]^T$$

then Eq. (A.5) can be written as

$$\sum_{i=1}^{n} x_i\mathbf{a}_i = \mathbf{b}$$

It follows from the above expression that Eq. (A.5) is solvable if and only if

$$\mathbf{b} \in \text{span}\{\mathbf{a}_1,\ \mathbf{a}_2,\ \ \ldots,\ \ \mathbf{a}_n\}$$

The subspace span$\{\mathbf{a}_1,\ \mathbf{a}_2,\ \ldots,\ \mathbf{a}_n\}$ is called the *range* of $\mathbf{A}$ and is denoted as $\mathcal{R}(\mathbf{A})$. Thus, Eq. (A.5) has a solution if and only if vector $\mathbf{b}$ is in the range of $\mathbf{A}$.

The dimension of $\mathcal{R}(\mathbf{A})$ is called the *rank* of $\mathbf{A}$, i.e., r = rank$(\mathbf{A})$ = dim$[\mathcal{R}(\mathbf{A})]$. Since $\mathbf{b} \in$ span$\{\mathbf{a}_1,\ \mathbf{a}_2,\ \ \ldots,\ \mathbf{a}_n\}$ is equivalent to

$$\text{span}\{\mathbf{b},\ \mathbf{a}_1,\ \ldots,\ \mathbf{a}_n\} = \text{span}\{\mathbf{a}_1,\ \mathbf{a}_2,\ \ \ldots,\ \ \mathbf{a}_n\}$$

we conclude that Eq. (A.5) is solvable if and only if

$$\text{rank}(\mathbf{A}) = \text{rank}([\mathbf{A}\ \mathbf{b}]) \tag{A.6}$$

It can be shown that rank$(\mathbf{A}) = $ rank$(\mathbf{A}^T)$. In other words, *the rank of a matrix is equal to the maximum number of linearly independent columns or rows*.

Another important concept associated with a matrix $\mathbf{A} \in R^{m\times n}$ is the *null space* of $\mathbf{A}$, which is defined as

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{x} : \ \mathbf{x} \in R^n, \ \mathbf{A}\mathbf{x} = \mathbf{0}\}$$

It can be readily verified that $\mathcal{N}(\mathbf{A})$ is a subspace of R^n. If $\mathbf{x}$ is a solution of Eq. (A.5) then $\mathbf{x} + \mathbf{z}$ with $\mathbf{z} \in \mathcal{N}(\mathbf{A})$ also satisfies Eq. (A.5). Hence Eq. (A.5) has a unique solution only if $\mathcal{N}(\mathbf{A})$ contains just one component, namely, the zero vector in R^n. Furthermore, it can be shown that for $\mathbf{A} \in R^{m\times n}$

$$\text{rank}(\mathbf{A}) + \text{dim}[\mathcal{N}(\mathbf{A})] = n \tag{A.7}$$

(see [2]). For the important special case where matrix $\mathbf{A}$ is square, i.e., $n = m$, the following statements are equivalent: (a) there exists a unique solution for Eq. (A.5); (b) $\mathcal{N}(\mathbf{A}) = \{\mathbf{0}\}$; (c) rank$(\mathbf{A}) = n$.

A matrix $\mathbf{A} \in R^{m\times n}$ is said to have full column rank if rank$(\mathbf{A}) = n$, i.e., the n column vectors of $\mathbf{A}$ are linearly independent, and $\mathbf{A}$ is said to have full row rank if rank$(\mathbf{A}) = m$, i.e., the m row vectors of $\mathbf{A}$ are linearly independent.

Example A.2 Find the rank and null space of matrix

$$\mathbf{V} = \begin{bmatrix} 1 & 0 & 3 & -1 \\ -1 & 2 & -7 & 5 \\ 3 & 1 & 7 & -1 \\ 0 & -1 & 2 & -2 \end{bmatrix}$$

Solution Note that the columns of $\mathbf{V}$ are the vectors $\mathbf{v}_i$ for $i = 1, 2, \ldots, 4$ in Example A.1. Since the maximum number of linearly independent columns is 2, we have rank$(\mathbf{V}) = 2$. To find $\mathcal{N}(\mathbf{V})$, we write $\mathbf{V} = [\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3 \ \mathbf{v}_4]$; hence the equation $\mathbf{V}\mathbf{x} = \mathbf{0}$ becomes

$$x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + x_3\mathbf{v}_3 + x_4\mathbf{v}_4 = \mathbf{0} \tag{A.8}$$

Using Eqs. (A.3) and (A.4), Eq. (A.8) can be expressed as

$$(x_1 + 3x_3 - x_4)\mathbf{v}_1 + (x_2 - 2x_3 + 2x_4)\mathbf{v}_2 = \mathbf{0}$$

which implies that

$$\begin{aligned} x_1 + 3x_3 - x_4 &= 0 \\ x_2 - 2x_3 + 2x_4 &= 0 \end{aligned}$$

i.e.,

$$\begin{aligned} x_1 &= -3x_3 + x_4 \\ x_2 &= 2x_3 - 2x_4 \end{aligned}$$

Hence any vector $\mathbf{x}$ that can be expressed as

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -3x_3 + x_4 \\ 2x_3 - 2x_4 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -3 \\ 2 \\ 1 \\ 0 \end{bmatrix} x_3 + \begin{bmatrix} 1 \\ -2 \\ 0 \\ 1 \end{bmatrix} x_4$$

with arbitrary x_3 and x_4 satisfies $\mathbf{Ax} = \mathbf{0}$. Since the two vectors in the above expression, namely,

$$\mathbf{n}_1 = \begin{bmatrix} -3 \\ 2 \\ 1 \\ 0 \end{bmatrix} \quad \text{and} \quad \mathbf{n}_2 = \begin{bmatrix} 1 \\ -2 \\ 0 \\ 1 \end{bmatrix}$$

are linearly independent, we have $\mathcal{N}(\mathbf{V}) = \text{span}\{\mathbf{n}_1,\ \mathbf{n}_2\}$.

■

A.4 Sherman-Morrison Formula

The Sherman-Morrison formula [4] states that given matrices $\mathbf{A} \in C^{n\times n}$, $\mathbf{U} \in C^{n\times p}$, $\mathbf{W} \in C^{p\times p}$, and $\mathbf{V} \in C^{n\times p}$, such that $\mathbf{A}^{-1}$, $\mathbf{W}^{-1}$ and $(\mathbf{W}^{-1} + \mathbf{V}^H\mathbf{A}^{-1}\mathbf{U})^{-1}$ exist, then the inverse of $\mathbf{A} + \mathbf{UWV}^H$ exists and is given by

$$(\mathbf{A} + \mathbf{UWV}^H)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{UY}^{-1}\mathbf{V}^H\mathbf{A}^{-1} \tag{A.9}$$

where

$$\mathbf{Y} = \mathbf{W}^{-1} + \mathbf{V}^H\mathbf{A}^{-1}\mathbf{U} \tag{A.10}$$

In particular, if $p = 1$ and $\mathbf{W} = 1$, then Eq. (A.9) assumes the form

$$(\mathbf{A} + \mathbf{uv}^H)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{uv}^H\mathbf{A}^{-1}}{1 + \mathbf{v}^H\mathbf{A}^{-1}\mathbf{u}} \tag{A.11}$$

where $\mathbf{u}$ and $\mathbf{v}$ are vectors in $C^{n\times 1}$. Eq. (A.11) is useful for computing the inverse of a rank-one modification of $\mathbf{A}$, namely, $\mathbf{A} + \mathbf{uv}^H$, if $\mathbf{A}^{-1}$ is available.

Example A.3 Find $\mathbf{A}^{-1}$ for

$$\mathbf{A} = \begin{bmatrix} 1.04 & 0.04 & \cdots & 0.04 \\ 0.04 & 1.04 & \cdots & 0.04 \\ \vdots & \vdots & & \vdots \\ 0.04 & 0.04 & \cdots & 1.04 \end{bmatrix} \in \mathcal{R}^{10\times 10}$$

Solution Matrix $\mathbf{A}$ can be treated as a rank-one perturbation of the identity matrix:

$$\mathbf{A} = \mathbf{I} + \mathbf{p}\mathbf{p}^T$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{p} = [0.2\ 0.2\ \cdots\ 0.2]^T$. Using Eq. (A.11), we can compute

$$\begin{aligned}\mathbf{A}^{-1} &= (\mathbf{I} + \mathbf{p}\mathbf{p}^T)^{-1} = \mathbf{I} - \frac{\mathbf{p}\mathbf{p}^T}{1 + \mathbf{p}^T\mathbf{p}} = \mathbf{I} - \frac{1}{1.4}\mathbf{p}\mathbf{p}^T \\ &= \begin{bmatrix} 0.9714 & -0.0286 & \cdots & -0.0286 \\ -0.0286 & 0.9714 & \cdots & -0.0286 \\ \vdots & \vdots & & \vdots \\ -0.0286 & -0.0286 & \ldots & 0.9714 \end{bmatrix}\end{aligned}$$

■

A.5 Eigenvalues and Eigenvectors

The *eigenvalues* of a matrix $\mathbf{A} \in C^{n\times n}$ are defined as the n roots of its so-called *characteristic equation*

$$\det(\lambda\mathbf{I} - \mathbf{A}) = 0 \tag{A.12}$$

If we denote the set of n eigenvalues $\{\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_n\}$ by $\lambda(\mathbf{A})$, then for a $\lambda_i \in \lambda(\mathbf{A})$, there exists a nonzero vector $\mathbf{x}_i \in C^{n\times 1}$ such that

$$\mathbf{A}\mathbf{x}_i = \lambda_i\mathbf{x}_i \tag{A.13}$$

Such a vector is called an *eigenvector* of $\mathbf{A}$ associated with eigenvalue λ_i.

Eigenvectors are not unique. For example, if $\mathbf{x}_i$ is an eigenvector of matrix $\mathbf{A}$ associated with eigenvalue λ_i and c is an arbitrary nonzero constant, then $c\mathbf{x}_i$ is also an eigenvector of $\mathbf{A}$ associated with eigenvalue λ_i.

If $\mathbf{A}$ has n distinct eigenvalues $\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_n$ with associated eigenvectors $\mathbf{x}_1,\ \mathbf{x}_2,\ \ldots,\ \mathbf{x}_n$, then these eigenvectors are linearly independent; hence we can write

$$\begin{aligned}\mathbf{A}[\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_n] &= [\mathbf{A}\mathbf{x}_1\ \mathbf{A}\mathbf{x}_2\ \cdots\ \mathbf{A}\mathbf{x}_n] = [\lambda_1\mathbf{x}_1\ \lambda_2\mathbf{x}_2\ \cdots\ \lambda_n\mathbf{x}_n] \\ &= [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_n]\begin{bmatrix} \lambda_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \lambda_n \end{bmatrix}\end{aligned}$$

In effect,

$$\mathbf{A}\mathbf{X} = \mathbf{X}\mathbf{\Lambda}$$

or

$$\mathbf{A} = \mathbf{X}\mathbf{\Lambda}\mathbf{X}^{-1} \tag{A.14}$$

with

$$\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_n] \qquad \text{and} \qquad \mathbf{\Lambda} = \text{diag}\{\lambda_1,\ \lambda_1,\ \ldots,\ \lambda_n\}$$

where diag$\{\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_n\}$ represents the diagonal matrix with components $\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_n$ along its diagonal. The relation in (A.14) is often referred to as an *eigendecomposition* of $\mathbf{A}$.

A concept that is closely related to the eigendecomposition in Eq. (A.14) is that of similarity transformation. Two square matrices $\mathbf{A}$ and $\mathbf{B}$ are said to be *similar* if there exists a nonsingular $\mathbf{X}$, called a *similarity transformation*, such that

$$\mathbf{A} = \mathbf{X}\mathbf{B}\mathbf{X}^{-1} \tag{A.15}$$

From Eq. (A.14), it follows that if the eigenvalues of $\mathbf{A}$ are distinct, then $\mathbf{A}$ is similar to $\mathbf{\Lambda} = \text{diag}\{\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_n\}$ and the similarity transformation involved, $\mathbf{X}$, is composed of the n eigenvectors of $\mathbf{A}$. For arbitrary matrices with repeated eigenvalues, the eigendecomposition becomes more complicated. The reader is referred to [1]–[3] for the theory and solution of the eigenvalue problem for the general case.

Example A.4 Find the diagonal matrix $\mathbf{\Lambda}$, if it exists, that is similar to matrix

$$\mathbf{A} = \begin{bmatrix} 4 & -3 & 1 & 1 \\ 2 & -1 & 1 & 1 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 2 & 1 \end{bmatrix}$$

Solution From Eq. (A.12), we have

$$\begin{aligned} \det(\lambda\mathbf{I} - \mathbf{A}) &= \det\begin{bmatrix} \lambda-4 & 3 \\ -2 & \lambda+1 \end{bmatrix} \cdot \det\begin{bmatrix} \lambda-1 & -2 \\ -2 & \lambda-1 \end{bmatrix} \\ &= (\lambda^2 - 3\lambda + 2)(\lambda^2 - 2\lambda - 3) \\ &= (\lambda-1)(\lambda-2)(\lambda+1)(\lambda-3) \end{aligned}$$

Hence the eigenvalues of $\mathbf{A}$ are $\lambda_1 = 1,\ \lambda_2 = 2,\ \lambda_3 = -1$, and $\lambda_4 = 3$. An eigenvector $\mathbf{x}_i$ associated with eigenvalue λ_i satisfies the relation

$$(\lambda_i\mathbf{I} - \mathbf{A})\mathbf{x}_i = \mathbf{0}$$

For $\lambda_1 = 1$, we have

$$\lambda_1\mathbf{I} - \mathbf{A} = \begin{bmatrix} -3 & 3 & -1 & -1 \\ -2 & 2 & -1 & -1 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 0 \end{bmatrix}$$

It is easy to verify that $\mathbf{x}_1 = [1\ 1\ 0\ 0]^T$ satisfies the relation

$$(\lambda_1 \mathbf{I} - \mathbf{A})\mathbf{x}_1 = \mathbf{0}$$

Similarly, $\mathbf{x}_2 = [3\ 2\ 0\ 0]^T$, $\mathbf{x}_3 = [0\ 0\ 1\ \ -1]^T$, and $\mathbf{x}_4 = [1\ 1\ 1\ 1]^T$ satisfy the relation

$$(\lambda_i \mathbf{I} - \mathbf{A})\mathbf{x}_i = \mathbf{0} \qquad \text{for } i = 2,\ 3,\ 4$$

If we let

$$\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3\ \mathbf{x}_4] = \begin{bmatrix} 1 & 3 & 0 & 1 \\ 1 & 2 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

then we have

$$\mathbf{AX} = \mathbf{\Lambda X}$$

where

$$\mathbf{\Lambda} = \text{diag}\{1,\ 2,\ -1,\ 3\}$$

■

A.6 Symmetric Matrices

The matrices encountered most frequently in numerical optimization are symmetric. For these matrices, an elegant eigendecomposition theory and corresponding computation methods are available. If $\mathbf{A} = \{a_{ij}\} \in R^{n\times n}$ is a symmetric matrix, i.e., $a_{ij} = a_{ji}$, then there exists an orthogonal matrix $\mathbf{X} \in R^{n\times n}$, i.e., $\mathbf{XX}^T = \mathbf{X}^T\mathbf{X} = \mathbf{I}_n$, such that

$$\mathbf{A} = \mathbf{X\Lambda X}^T \tag{A.16}$$

where $\mathbf{\Lambda} = \text{diag}\{\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_n\}$. If $\mathbf{A} \in C^{n\times n}$ is such that $\mathbf{A} = \mathbf{A}^H$, then $\mathbf{A}$ is referred to as a *Hermitian matrix*. In such a case, there exists a so-called *unitary matrix* $\mathbf{U} \in C^{n\times n}$ for which $\mathbf{UU}^H = \mathbf{U}^H\mathbf{U} = \mathbf{I}_n$ such that

$$\mathbf{A} = \mathbf{U\Lambda U}^H \tag{A.17}$$

In Eqs. (A.16) and (A.17), the diagonal components of $\mathbf{\Lambda}$ are eigenvalues of $\mathbf{A}$, and the columns of $\mathbf{X}$ and $\mathbf{U}$ are corresponding eigenvectors of $\mathbf{A}$.

The following properties can be readily verified:

(a) A square matrix is nonsingular if and only if all its eigenvalues are nonzero.

(b) The magnitudes of the eigenvalues of an orthogonal or unitary matrix are always equal to unity.

(c) The eigenvalues of a symmetric or Hermitian matrix are always real.

(d) The determinant of a square matrix is equal to the product of its eigenvalues.

A symmetric matrix $\mathbf{A} \in R^{n\times n}$ is said to be *positive definite, positive semidefinite, negative semidefinite, negative definite* if $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0,\ \mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0,\ \mathbf{x}^T\mathbf{A}\mathbf{x} \leq 0,\ \mathbf{x}^T\mathbf{A}\mathbf{x} < 0$, respectively, for all nonzero $\mathbf{x} \in R^{n\times 1}$.

Using the decomposition in Eq. (A.16), it can be shown that matrix $\mathbf{A}$ is positive definite, positive semidefinite, negative semidefinite, negative definite, if and only if its eigenvalues are positive, nonnegative, nonpositive, negative, respectively. Otherwise, $\mathbf{A}$ is said to be indefinite. We use the shorthand notation $\mathbf{A} \succ,\ \succeq,\ \preceq,\ \prec \mathbf{0}$ to indicate that $\mathbf{A}$ is positive definite, positive semidefinite, negative semidefinite, negative definite throughout the book.

Another approach for the characterization of a square matrix $\mathbf{A}$ is based on the evaluation of the *leading principal minor determinants*. A *minor determinant*, which is usually referred to as a *minor*, is the determinant of a submatrix obtained by deleting a number of rows and an equal number of columns from the matrix. Specifically, a minor of order r of an $n \times n$ matrix $\mathbf{A}$ is obtained by deleting $n - r$ rows and $n - r$ columns. For example, if

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

then

$$\Delta_3^{(123,123)} = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}, \quad \Delta_3^{(134,124)} = \begin{vmatrix} a_{11} & a_{12} & a_{14} \\ a_{31} & a_{32} & a_{34} \\ a_{41} & a_{42} & a_{44} \end{vmatrix}$$

and

$$\Delta_2^{(12,12)} = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \quad \Delta_2^{(13,14)} = \begin{vmatrix} a_{11} & a_{14} \\ a_{31} & a_{34} \end{vmatrix}$$

$$\Delta_2^{(24,13)} = \begin{vmatrix} a_{21} & a_{23} \\ a_{41} & a_{43} \end{vmatrix}, \quad \Delta_2^{(34,34)} = \begin{vmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{vmatrix}$$

are third-order and second-order minors, respectively. An nth-order minor is the determinant of the matrix itself and a first-order minor, i.e., if $n-1$ rows and $n - 1$ columns are deleted, is simply the value of a single matrix component.[1]

If the indices of the deleted rows are the same as those of the deleted columns, then the minor is said to be a *principal minor*, e.g., $\Delta_3^{(123,123)}$, $\Delta_2^{(12,12)}$, and $\Delta_2^{(34,34)}$ in the above examples.

[1]The zeroth-order minor is often defined to be unity.

Principal minors $\Delta_3^{(123,123)}$ and $\Delta_2^{(12,12)}$ in the above examples can be represented by

$$\Delta_3^{(1,2,3)} = \det \mathbf{H}_3^{(1,2,3)}$$

and

$$\Delta_2^{(1,2)} = \det \mathbf{H}_2^{(1,2)}$$

respectively. An arbitrary principal minor of order i can be represented by

$$\Delta_i^{(l)} = \det \mathbf{H}_i^{(l)}$$

where

$$\mathbf{H}_i^{(l)} = \begin{bmatrix} a_{l_1 l_1} & a_{l_1 l_2} & \cdots & a_{l_1 l_i} \\ a_{l_2 l_1} & a_{l_2 l_2} & \cdots & a_{l_2 l_i} \\ \vdots & \vdots & & \vdots \\ a_{l_i l_1} & a_{l_i l_2} & \cdots & a_{l_i l_i} \end{bmatrix}$$

and $l \in \{l_1,\ l_2,\ \ldots,\ l_i\}$ with $1 \leq l_1 < l_2 < \cdots < l_i \leq n$ is the set of rows (and columns) retained in submatrix $\mathbf{H}_i^{(l)}$.

The specific principal minors

$$\Delta_r = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1r} \\ a_{21} & a_{22} & \cdots & a_{2r} \\ \vdots & \vdots & & \vdots \\ a_{r1} & a_{r2} & \cdots & a_{rr} \end{vmatrix} = \det \mathbf{H}_r$$

for $1 \leq r \leq n$ are said to be the *leading principal minors* of an $n \times n$ matrix. For a 4×4 matrix, the complete set of leading principal minors is as follows:

$$\Delta_1 = a_{11}, \quad \Delta_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$$

$$\Delta_3 = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}, \quad \Delta_4 = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{vmatrix}$$

The leading principal minors of a matrix $\mathbf{A}$ or its negative $-\mathbf{A}$ can be used to establish whether the matrix is positive or negative definite whereas the principal minors of $\mathbf{A}$ or $-\mathbf{A}$ can be used to establish whether the matrix is positive or negative semidefinite. These principles are stated in terms of Theorem 2.9 in Chap. 2 and are often used to establish the nature of the Hessian matrix in optimization algorithms.

The fact that a nonnegative real number has positive and negative square roots can be extended to the class of positive semidefinite matrices. Assuming that matrix $\mathbf{A} \in R^{n\times n}$ is positive semidefinite, we can write its eigendecomposition in Eq. (A.16) as

$$\mathbf{A} = \mathbf{X\Lambda X}^T = \mathbf{X\Lambda}^{1/2}\mathbf{WW}^T\mathbf{\Lambda}^{1/2}\mathbf{X}^T$$

where $\mathbf{\Lambda}^{1/2} = \text{diag}\{\lambda_1^{1/2},\ \lambda_2^{1/2},\ \ldots,\ \lambda_n^{1/2}\}$ and $\mathbf{W}$ is an arbitrary orthogonal matrix, which leads to

$$\mathbf{A} = \mathbf{A}^{1/2}(\mathbf{A}^{1/2})^T \tag{A.18}$$

where $\mathbf{A}^{1/2} = \mathbf{X\Lambda}^{1/2}\mathbf{W}$ and is called an *asymmetric square root* of $\mathbf{A}$. Since matrix $\mathbf{W}$ can be an arbitrary orthogonal matrix, an infinite number of asymmetric square roots of $\mathbf{A}$ exist. Alternatively, since $\mathbf{X}$ is an orthogonal matrix, we can write

$$\mathbf{A} = (\alpha\mathbf{X\Lambda}^{1/2}\mathbf{X}^T)(\alpha\mathbf{X\Lambda}^{1/2}\mathbf{X}^T)$$

where α is either 1 or -1, which gives

$$\mathbf{A} = \mathbf{A}^{1/2}\mathbf{A}^{1/2} \tag{A.19}$$

where $\mathbf{A}^{1/2} = \alpha\mathbf{X\Lambda}^{1/2}\mathbf{X}^T$ and is called a *symmetric square root* of $\mathbf{A}$. Again, because α can be either 1 or -1, more than one symmetric square roots exist. Obviously, the symmetric square roots $\mathbf{X\Lambda}^{1/2}\mathbf{X}^T$ and $-\mathbf{X\Lambda}^{1/2}\mathbf{X}^T$ are positive semidefinite and negative semidefinite, respectively.

If $\mathbf{A}$ is a complex-valued positive semidefinite matrix, then *non-Hermitian* and *Hermitian square roots* of $\mathbf{A}$ can be obtained using the eigendecomposition in Eq. (A.17). For example, we can write

$$\mathbf{A} = \mathbf{A}^{1/2}(\mathbf{A}^{1/2})^H$$

where $\mathbf{A}^{1/2} = \mathbf{U\Lambda}^{1/2}\mathbf{W}$ is a non-Hermitian square root of $\mathbf{A}$ if $\mathbf{W}$ is unitary. On the other hand,

$$\mathbf{A} = \mathbf{A}^{1/2}\mathbf{A}^{1/2}$$

where $\mathbf{A}^{1/2} = \alpha\mathbf{U\Lambda}^{1/2}\mathbf{U}^H$ is a Hermitian square root if $\alpha = 1$ or $\alpha = -1$.

Example A.5 Verify that

$$\mathbf{A} = \begin{bmatrix} 2.5 & 0 & 1.5 \\ 0 & \sqrt{2} & 0 \\ 1.5 & 0 & 2.5 \end{bmatrix}$$

is positive definite and compute a symmetric square root of $\mathbf{A}$.

Solution An eigendecomposition of matrix $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{X\Lambda X}^T$$

with

$$\mathbf{\Lambda} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{X} = \begin{bmatrix} \sqrt{2}/2 & 0 & -\sqrt{2}/2 \\ 0 & -1 & 0 \\ \sqrt{2}/2 & 0 & \sqrt{2}/2 \end{bmatrix}$$

Since the eigenvalues of $\mathbf{A}$ are all positive, $\mathbf{A}$ is positive definite. A symmetric square root of $\mathbf{A}$ is given by

$$\mathbf{A}^{1/2} = \mathbf{X}\mathbf{\Lambda}^{1/2}\mathbf{X}^T = \begin{bmatrix} 1.5 & 0 & 0.5 \\ 0 & \sqrt{2} & 0 \\ 0.5 & 0 & 1.5 \end{bmatrix}$$

■

A.7 Trace

The trace of an $n \times n$ square matrix, $\mathbf{A} = \{a_{ij}\}$, is *the sum of its diagonal components*, i.e.,

$$\text{trace}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}$$

It can be verified that the trace of a square matrix $\mathbf{A}$ with eigenvalues $\lambda_1,\ \lambda_2,\ \ldots,\ \lambda_n$ is equal to the sum of its eigenvalues, i.e.,

$$\text{trace}(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i$$

A useful property pertaining to the product of two matrices is that the trace of a square matrix $\mathbf{AB}$ is equal to the trace of matrix $\mathbf{BA}$, i.e.,

$$\text{trace}(\mathbf{AB}) = \text{trace}(\mathbf{BA}) \tag{A.20}$$

By applying Eq. (A.20) to the quadratic form $\mathbf{x}^T\mathbf{H}\mathbf{x}$, we obtain

$$\mathbf{x}^T\mathbf{H}\mathbf{x} = \text{trace}(\mathbf{x}^T\mathbf{H}\mathbf{x}) = \text{trace}(\mathbf{H}\mathbf{x}\mathbf{x}^T) = \text{trace}(\mathbf{H}\mathbf{X})$$

where $\mathbf{X} = \mathbf{x}\mathbf{x}^T$. Moreover, we can write a general quadratic function as

$$\mathbf{x}^T\mathbf{H}\mathbf{x} + 2\mathbf{p}^T\mathbf{x} + \kappa = \text{trace}(\hat{\mathbf{H}}\hat{\mathbf{X}}) \tag{A.21}$$

where

$$\hat{\mathbf{H}} = \begin{bmatrix} \mathbf{H} & \mathbf{p} \\ \mathbf{p}^T & \kappa \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{X}} = \begin{bmatrix} \mathbf{x}\mathbf{x}^T & \mathbf{x} \\ \mathbf{x}^T & 1 \end{bmatrix}$$

A.8 Vector Norms and Matrix Norms

A.8.1 Vector norms

The L_p norm of a vector $\mathbf{x} \in C^n$ for $p \geq 1$ is given by

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{A.22}$$

where p is a positive integer and x_i is the ith component of $\mathbf{x}$. The most popular L_p norms are $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_\infty$, where the infinity norm $\|\cdot\|_\infty$ can easily be shown to satisfy the relation

$$\|\mathbf{x}\|_\infty = \lim_{p\to\infty} \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} = \max_i |x_i| \tag{A.23}$$

For example, if $\mathbf{x} = [1\ 2\ \cdots\ 100]^T$, then $\|\mathbf{x}\| = 581.68$, $\|\mathbf{x}\|_{10} = 125.38$, $\|\mathbf{x}\|_{50} = 101.85$, $\|\mathbf{x}\|_{100} = 100.45$, $\|\mathbf{x}\|_{200} = 100.07$ and, of course, $\|\mathbf{x}\|_\infty = 100$.

The important point to note here is that for an even p, the L_p norm of a vector is a *differentiable* function of its components but the L_∞ norm is *not*. So when the L_∞ norm is used in a design problem, we can replace it by an L_p norm (with p even) so that powerful calculus-based tools can be used to solve the problem. Obviously, the results obtained can only be *approximate* with respect to the original design problem. However, as indicated by Eq. (9.23), the difference between the approximate and exact solutions becomes insignificant if p is sufficiently large.

The inner product of two vectors $\mathbf{x},\ \mathbf{y} \in C^n$ is a scalar given by

$$\mathbf{x}^H \mathbf{y} = \sum_{i=1}^{n} x_i^* y_i$$

where x_i^* denotes the complex-conjugate of x_i. Frequently, we need to estimate the absolute value of $\mathbf{x}^H\mathbf{y}$. There are two well-known inequalities that provide tight upper bounds for $|\mathbf{x}^H\mathbf{y}|$, namely, the *Hölder inequality*

$$|\mathbf{x}^H \mathbf{y}| \leq \|\mathbf{x}\|_p \|\mathbf{y}\|_q \tag{A.24}$$

which holds for any $p \geq 1$ and $q \geq 1$ satisfying the equality

$$\frac{1}{p} + \frac{1}{q} = 1$$

and the *Cauchy-Schwartz inequality* which is the special case of the Hölder inequality with $p = q = 2$, i.e.,

$$|\mathbf{x}^H\mathbf{y}| \le \|\mathbf{x}\|_2\|\mathbf{y}\|_2 \tag{A.25}$$

If vectors $\mathbf{x}$ and $\mathbf{y}$ have unity lengths, i.e., $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = 1$, then Eq. (A.25) becomes

$$|\mathbf{x}^H\mathbf{y}| \le 1 \tag{A.26}$$

A geometric interpretation of Eq. (A.26) is that for unit vectors $\mathbf{x}$ and $\mathbf{y}$, the inner product $\mathbf{x}^H\mathbf{y}$ is equal to $\cos\theta$, where θ denotes the angle between the two vectors, whose absolute value is always less than one.

Another property of the L_2 norm is its invariance under orthogonal or unitary transformation. That is, if $\mathbf{A}$ is an orthogonal or unitary matrix, then

$$\|\mathbf{A}\mathbf{x}\|_2 = \|\mathbf{x}\|_2 \tag{A.27}$$

The L_p norm of a vector $\mathbf{x}$, $\|\mathbf{x}\|_p$, is monotonically decreasing with respect to p for $p \ge 1$. For example, we can relate $\|\mathbf{x}\|_1$ and $\|\mathbf{x}\|_2$ as

$$\begin{aligned}\|\mathbf{x}\|_1^2 &= \left(\sum_{i=1}^{n}|x_i|\right)^2 \\ &= |x_1|^2 + |x_2|^2 + \cdots + |x_n|^2 + 2|x_1x_2| + \cdots + 2|x_{n-1}x_n| \\ &\ge |x_1|^2 + |x_2|^2 + \cdots + |x_n|^2 = \|\mathbf{x}\|_2^2\end{aligned}$$

which implies that

$$\|\mathbf{x}\|_1 \ge \|\mathbf{x}\|_2$$

Furthermore, if $\|\mathbf{x}\|_\infty$ is numerically equal to $|x_k|$ for some index k, i.e.,

$$\|\mathbf{x}\|_\infty = \max_i |x_i| = |x_k|$$

then we can write

$$\|\mathbf{x}\|_2 = (|x_1|^2 + \cdots + |x_n|^2)^{1/2} \ge (|x_k|^2)^{1/2} = |x_k| = \|\mathbf{x}\|_\infty$$

i.e.,

$$\|\mathbf{x}\|_2 \ge \|\mathbf{x}\|_\infty$$

Therefore, we have

$$\|\mathbf{x}\|_1 \ge \|\mathbf{x}\|_2 \ge \|\mathbf{x}\|_\infty$$

In general, it can be shown that

$$\|\mathbf{x}\|_1 \ge \|\mathbf{x}\|_2 \ge \|\mathbf{x}\|_3 \ge \cdots \ge \|\mathbf{x}\|_\infty$$

A.8.2 Matrix norms

The L_p norm of matrix $\mathbf{A} = \{a_{ij}\} \in C^{m\times n}$ is defined as

$$\|\mathbf{A}\|_p = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|_p}{\|\mathbf{x}\|_p} \qquad \text{for } p \geq 1 \tag{A.28}$$

The most useful matrix L_p norm is the L_2 norm

$$\|\mathbf{A}\|_2 = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \left[\max_i \left|\lambda_i(\mathbf{A}^H\mathbf{A})\right|\right]^{1/2} = \left[\max_i \left|\lambda_i(\mathbf{A}\mathbf{A}^H)\right|\right]^{1/2} \tag{A.29}$$

which can be easily computed as the square root of the largest eigenvalue magnitude in $\mathbf{A}^H\mathbf{A}$ or $\mathbf{A}\mathbf{A}^H$. Some other frequently used matrix L_p norms are

$$\|\mathbf{A}\|_1 = \max_{\mathbf{x}\neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_1}{\|\mathbf{x}\|_1} = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|$$

and

$$\|\mathbf{A}\|_\infty = \max_{x\neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|$$

Another popular matrix norm is the Frobenius norm which is defined as

$$\|\mathbf{A}\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2\right)^{1/2} \tag{A.30}$$

which can also be calculated as

$$\|\mathbf{A}\|_F = [\text{trace}(\mathbf{A}^H\mathbf{A})]^{1/2} = [\text{trace}(\mathbf{A}\mathbf{A}^H)]^{1/2} \tag{A.31}$$

Note that the matrix L_2 norm and the Frobenius norm are *invariant* under orthogonal or unitary transformation, i.e., if $\mathbf{U} \in C^{n\times n}$ and $\mathbf{V} \in C^{m\times m}$ are unitary or orthogonal matrices, then

$$\|\mathbf{U}\mathbf{A}\mathbf{V}\|_2 = \|\mathbf{A}\|_2 \tag{A.32}$$

and

$$\|\mathbf{U}\mathbf{A}\mathbf{V}\|_F = \|\mathbf{A}\|_F \tag{A.33}$$

Example A.6 Evaluate matrix norms $||\mathbf{A}||_1$, $||\mathbf{A}||_2$, $||\mathbf{A}||_\infty$, and $||\mathbf{A}||_F$ for

$$\mathbf{A} = \begin{bmatrix} 1 & 5 & 6 & 3 \\ 0 & 4 & -7 & 0 \\ 3 & 1 & 4 & 1 \\ -1 & 1 & 0 & 1 \end{bmatrix}$$

Solution

$$||\mathbf{A}||_1 = \max_{1\le j\le 4}\left(\sum_{i=1}^{4}|a_{ij}|\right) = \max\{5,\ 11,\ 17,\ 5\} = 17$$

$$||\mathbf{A}||_\infty = \max_{1\le i\le 4}\left(\sum_{j=1}^{4}|a_{ij}|\right) = \max\{15,\ 11,\ 9,\ 3\} = 15$$

$$||\mathbf{A}||_F = \left(\sum_{i=1}^{4}\sum_{j=1}^{4}|a_{ij}|^2\right)^{1/2} = \sqrt{166} = 12.8841$$

To obtain $||\mathbf{A}||_2$, we compute the eigenvalues of $\mathbf{A}^T\mathbf{A}$ as

$$\lambda(\mathbf{A}^T\mathbf{A}) = \{0.2099,\ 6.9877,\ 47.4010,\ 111.4014\}$$

Hence

$$||\mathbf{A}||_2 = [\max_i |\lambda_i(\mathbf{A}^T\mathbf{A})|]^{1/2} = \sqrt{111.4014} = 10.5547$$

■

A.9 Singular-Value Decomposition

Given a matrix $\mathbf{A} \in C^{m\times n}$ of rank r, there exist unitary matrices $\mathbf{U} \in C^{m\times m}$ and $\mathbf{V} \in C^{n\times n}$ such that

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^H \tag{A.34a}$$

where

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{m\times n} \tag{A.34b}$$

and

$$\mathbf{S} = \text{diag}\{\sigma_1,\ \sigma_2,\ \ldots,\ \sigma_r\} \tag{A.34c}$$

with $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_r > 0$.

The matrix decomposition in Eq. (A.34a) is known as the *singular-value* decomposition (SVD) of $\mathbf{A}$. It has many applications in optimization and elsewhere. If $\mathbf{A}$ is a real-valued matrix, then $\mathbf{U}$ and $\mathbf{V}$ in Eq. (A.34a) become orthogonal matrices and $\mathbf{V}^H$ becomes $\mathbf{V}^T$. The positive scalars σ_i for $i = 1,\ 2,\ \ldots,\ r$ in Eq. (A.34c) are called the *singular values* of $\mathbf{A}$. If $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_m]$ and $\mathbf{V} = [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_n]$, vectors $\mathbf{u}_i$ and $\mathbf{v}_i$ are called the *left* and *right singular vectors* of $\mathbf{A}$, respectively. From Eq. (A.34), it follows that

$$\mathbf{A}\mathbf{A}^H = \mathbf{U}\begin{bmatrix} \mathbf{S}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}_{m\times m}\mathbf{U}^H \tag{A.35a}$$

and

$$\mathbf{A}^H\mathbf{A} = \mathbf{V}\begin{bmatrix}\mathbf{S}^2 & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}_{n\times n}\mathbf{V}^H \tag{A.35b}$$

Therefore, the singular values of $\mathbf{A}$ are the positive square roots of the nonzero eigenvalues of $\mathbf{AA}^H$ (or $\mathbf{A}^H\mathbf{A}$), the ith left singular vector $\mathbf{u}_i$ is the ith eigenvector of $\mathbf{AA}^H$, and the ith right singular vector $\mathbf{v}_i$ is the ith eigenvector of $\mathbf{A}^H\mathbf{A}$.

Several important applications of the SVD are as follows:

(a) The L_2 norm and Frobenius norm of a matrix $\mathbf{A} \in C^{m\times n}$ of rank r are given, respectively, by

$$\|\mathbf{A}\|_2 = \sigma_1 \tag{A.36}$$

and

$$\|\mathbf{A}\|_F = \left(\sum_{i=1}^{r}\sigma_i^2\right)^{1/2} \tag{A.37}$$

(b) The *condition number* of a nonsingular matrix $\mathbf{A} \in C^{n\times n}$ is defined as

$$\text{cond}(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^{-1}\|_2 = \frac{\sigma_1}{\sigma_n} \tag{A.38}$$

(c) The range and null space of a matrix $\mathbf{A} \in C^{m\times n}$ of rank r assume the forms

$$\mathcal{R}(\mathbf{A}) = \text{span}\{\mathbf{u}_1,\ \mathbf{u}_2,\ \ldots,\ \mathbf{u}_r\} \tag{A.39}$$

$$\mathcal{N}(\mathbf{A}) = \text{span}\{\mathbf{v}_{r+1},\ \mathbf{v}_{r+2},\ \ldots,\ \mathbf{v}_n\} \tag{A.40}$$

(d) Properties and computation of Moore-Penrose pseudo-inverse:

The Moore-Penrose pseudo-inverse of a matrix $\mathbf{A} \in C^{m\times n}$ is defined as the matrix $\mathbf{A}^+ \in C^{n\times m}$ that satisfies the following four conditions:

(i) $\mathbf{AA}^+\mathbf{A} = \mathbf{A}$

(ii) $\mathbf{A}^+\mathbf{AA}^+ = \mathbf{A}^+$

(iii) $(\mathbf{AA}^+)^H = \mathbf{AA}^+$

(iv) $(\mathbf{A}^+\mathbf{A})^H = \mathbf{A}^+\mathbf{A}$

Using the SVD of $\mathbf{A}$ in Eq. (A.34), the Moore-Penrose pseudo-inverse of $\mathbf{A}$ can be obtained as

$$\mathbf{A}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^H \tag{A.41a}$$

where

$$\mathbf{\Sigma}^+ = \begin{bmatrix}\mathbf{S}^{-1} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}_{n\times m} \tag{A.41b}$$

and

$$\mathbf{S}^{-1} = \text{diag}\{\sigma_1^{-1},\ \sigma_2^{-1},\ \ldots,\ \sigma_r^{-1}\} \tag{A.41c}$$

Consequently, we have

$$\mathbf{A}^+ = \sum_{i=1}^{r} \frac{\mathbf{v}_i \mathbf{u}_i^H}{\sigma_i} \tag{A.42}$$

(e) For an underdetermined system of linear equations

$$\mathbf{Ax} = \mathbf{b} \tag{A.43}$$

where $\mathbf{A} \in C^{m\times n}$, $\mathbf{b} \in C^{m\times 1}$ with $m < n$, and $\mathbf{b} \in \mathcal{R}(\mathbf{A})$, all the solutions of Eq. (A.43) are characterized by

$$\mathbf{x} = \mathbf{A}^+\mathbf{b} + \mathbf{V}_r\boldsymbol{\phi} \tag{A.44a}$$

where $\mathbf{A}^+$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}$,

$$\mathbf{V}_r = [\mathbf{v}_{r+1}\ \mathbf{v}_{r+2}\ \cdots\ \mathbf{v}_n] \tag{A.44b}$$

is a matrix of dimension $n \times (n-r)$ composed of the last $n-r$ columns of matrix $\mathbf{V}$ which is obtained by constructing the SVD of $\mathbf{A}$ in Eq. (A.34), and $\boldsymbol{\phi} \in C^{(n-r)\times 1}$ is an *arbitrary* $(n-r)$-dimensional vector. Note that the first term in Eq. (A.44a), i.e., $\mathbf{A}^+\mathbf{b}$, is a solution of Eq. (A.43) while the second term, $\mathbf{V}_r\boldsymbol{\phi}$, belongs to the null space of $\mathbf{A}$ (see Eq. (A.40)). Through vector $\boldsymbol{\phi}$, the expression in Eq. (A.44) parameterizes all the solutions of an underdetermined system of linear equations.

Example A.7 Perform the SVD of matrix

$$\mathbf{A} = \begin{bmatrix} 2.8284 & -1 & 1 \\ 2.8284 & 1 & -1 \end{bmatrix}$$

and compute $||\mathbf{A}||_2$, $||\mathbf{A}||_F$, and $\mathbf{A}^+$.

Solution To compute matrix $\mathbf{V}$ in Eq. (A.34a), from Eq. (A.35b) we obtain

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 16 & 0 & 0 \\ 0 & 2 & -2 \\ 0 & -2 & 2 \end{bmatrix} = \mathbf{V} \begin{bmatrix} 16 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{V}^T$$

where

$$\mathbf{V} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.7071 & -0.7071 \\ 0 & -0.7071 & -0.7071 \end{bmatrix} = [\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3]$$

Hence the nonzero singular values of $\mathbf{A}$ are $\sigma_1 = \sqrt{16} = 4$ and $\sigma_2 = \sqrt{4} = 2$. Now we can write (A.34a) as $\mathbf{U\Sigma} = \mathbf{AV}$, where

$$\mathbf{U\Sigma} = [\sigma_1\mathbf{u}_1 \;\; \sigma_2\mathbf{u}_2 \;\; \mathbf{0}] = [4\mathbf{u}_1 \;\; 2\mathbf{u}_2 \;\; \mathbf{0}]$$

and

$$\mathbf{AV} = \begin{bmatrix} 2.8284 & -1.4142 & 0 \\ 2.8284 & 1.4142 & 0 \end{bmatrix}$$

Hence

$$\mathbf{u}_1 = \frac{1}{4}\begin{bmatrix} 2.8284 \\ 2.8284 \end{bmatrix} = \begin{bmatrix} 0.7071 \\ 0.7071 \end{bmatrix}, \quad \mathbf{u}_2 = \frac{1}{2}\begin{bmatrix} -1.4142 \\ 1.4142 \end{bmatrix} = \begin{bmatrix} -0.7071 \\ 0.7071 \end{bmatrix}$$

and

$$\mathbf{U} = [\mathbf{u}_1 \; \mathbf{u}_2] = \begin{bmatrix} 0.7071 & -0.7071 \\ 0.7071 & 0.7071 \end{bmatrix}$$

On using Eqs. (A.36) and (A.37), we have

$$||\mathbf{A}||_2 = \sigma_1 = 4 \quad \text{and} \quad ||\mathbf{A}||_F = (\sigma_1^2 + \sigma_2^2)^{1/2} = \sqrt{20} = 4.4721$$

Now from Eq. (A.42), we obtain

$$\mathbf{A}^+ = \frac{\mathbf{v}_1\mathbf{u}_1^T}{\sigma_1} + \frac{\mathbf{v}_2\mathbf{u}_2^T}{\sigma_2} = \begin{bmatrix} 0.1768 & 0.1768 \\ -0.2500 & 0.2500 \\ 0.2500 & -0.2500 \end{bmatrix}$$

■

A.10 Orthogonal Projections

Let $\mathcal{S}$ be a subspace in C^n. Matrix $\mathbf{P} \in C^{n\times n}$ is said to be an orthogonal projection matrix onto $\mathcal{S}$ if $\mathcal{R}(\mathbf{P}) = \mathcal{S}$, $\mathbf{P}^2 = \mathbf{P}$, and $\mathbf{P}^H = \mathbf{P}$, where $\mathcal{R}(\mathbf{P})$ denotes the range of transformation $\mathbf{P}$ (see Sec. A.3), i.e., $\mathcal{R}(\mathbf{P}) = \{\mathbf{y} : \mathbf{y} = \mathbf{Px}, \ \mathbf{x} \in C^n\}$. The term 'orthogonal projection' originates from the fact that if $\mathbf{x} \in C^n$ is a vector outside $\mathcal{S}$, then $\mathbf{Px}$ is a vector in $\mathcal{S}$ such that $\mathbf{x} - \mathbf{Px}$ is orthogonal to every vector in $\mathcal{S}$ and $\|\mathbf{x} - \mathbf{Px}\|$ is the minimum distance between $\mathbf{x}$ and $\mathbf{s}$, i.e., $\min \|\mathbf{x} - \mathbf{s}\|$, for $\mathbf{s} \in \mathcal{S}$, as illustrated in Fig. A.1.

Let $\{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k\}$ be a basis of a subspace $\mathcal{S}$ of dimension k (see Sec. A.2) such that $||\mathbf{s}_i|| = 1$ and $\mathbf{s}_i^T\mathbf{s}_j = 0$ for $i, \ j = 1, \ 2, \ \ldots, k$ and $i \neq j$. Such a basis is called *orthonormal*. It can be readily verified that an orthogonal projection matrix onto $\mathcal{S}$ can be explicitly constructed in terms of an orthonormal basis as

$$\mathbf{P} = \mathbf{SS}^H \tag{A.45a}$$

where

$$\mathbf{S} = [\mathbf{s}_1 \; \mathbf{s}_2 \; \cdots \; \mathbf{s}_k] \tag{A.45b}$$

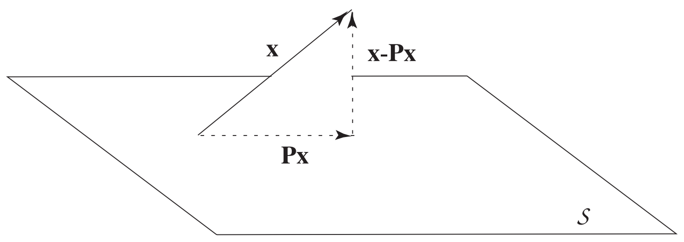


Figure A.1. Orthogonal projection of $\mathbf{x}$ onto subspace $\mathcal{S}$.

It follows from Eqs. (A.39), (A.40), and (A.45) that $[\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_r]\cdot[\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_r]^H$ is the orthogonal projection onto $\mathcal{R}(\mathbf{A})$ and $[\mathbf{v}_{r+1}\ \mathbf{v}_{r+2}\ \cdots\ \mathbf{v}_n]\cdot[\mathbf{v}_{r+1}\ \mathbf{v}_{r+2}\ \cdots\ \mathbf{v}_n]^H$ is the orthogonal projection onto $\mathcal{N}(\mathbf{A})$.

Example A.8 Let $\mathcal{S} = \text{span}\{\mathbf{v}_1,\ \mathbf{v}_2\}$ where

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{v}_2 = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$$

Find the orthogonal projection onto $\mathcal{S}$.

Solution First, we need to find an orthonormal basis $\{\mathbf{s}_1,\ \mathbf{s}_2\}$ of subspace $\mathcal{S}$. To this end, we take

$$\mathbf{s}_1 = \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|} = \begin{bmatrix} 1/\sqrt{3} \\ 1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix}$$

Then we try to find vector $\hat{\mathbf{s}}_2$ such that $\hat{\mathbf{s}}_2 \in \mathcal{S}$ and $\hat{\mathbf{s}}_2$ is orthogonal to $\mathbf{s}_1$. Such an $\hat{\mathbf{s}}_2$ must satisfy the relation

$$\hat{\mathbf{s}}_2 = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2$$

for some $\alpha_1,\ \alpha_2$ and

$$\hat{\mathbf{s}}_2^T \mathbf{s}_1 = 0$$

Hence we have

$$(\alpha_1 \mathbf{v}_1^T + \alpha_2 \mathbf{v}_2^T)\mathbf{s}_1 = \alpha_1 \mathbf{v}_1^T \mathbf{s}_1 + \alpha_2 \mathbf{v}_2^T \mathbf{s}_1 = \sqrt{3}\alpha_1 + \frac{1}{\sqrt{3}}\alpha_2 = 0$$

i.e., $\alpha_2 = -3\alpha_1$. Thus

$$\hat{\mathbf{s}}_2 = \alpha_1 \mathbf{v}_1 - 3\alpha_1 \mathbf{v}_2 = \alpha_1 \begin{bmatrix} 4 \\ -2 \\ -2 \end{bmatrix}$$

where α_1 is a parameter that can assume an arbitrary nonzero value.

By normalizing vector $\hat{\mathbf{s}}_2$, we obtain

$$\mathbf{s}_2 = \frac{\hat{\mathbf{s}}_2}{\|\hat{\mathbf{s}}_2\|} = \frac{1}{\sqrt{4^2+(-2)^2+(-2)^2}}\begin{bmatrix} 4 \\ -2 \\ -2 \end{bmatrix} = \frac{1}{\sqrt{6}}\begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix}$$

It now follows from Eq. (A.45) that the orthogonal projection onto $\mathcal{S}$ can be characterized by

$$\mathbf{P} = [\mathbf{s}_1\ \mathbf{s}_2][\mathbf{s}_1\ \mathbf{s}_2]^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 \end{bmatrix}$$

■

A.11 Householder Transformations and Givens Rotations

A.11.1 Householder transformations

The *Householder transformation* associated with a nonzero vector $\mathbf{u} \in R^{n\times 1}$ is characterized by the symmetric orthogonal matrix

$$\mathbf{H} = \mathbf{I} - 2\frac{\mathbf{u}\mathbf{u}^T}{\|\mathbf{u}\|^2} \tag{A.46}$$

If

$$\mathbf{u} = \mathbf{x} - \|\mathbf{x}\|\mathbf{e}_1 \tag{A.47}$$

where $\mathbf{e}_1 = [1\ 0\ \cdots\ 0]^T$, then the Householder transformation will convert vector $\mathbf{x}$ to coordinate vector $\mathbf{e}_1$ to within a scale factor $\|\mathbf{x}\|$, i.e.,

$$\mathbf{H}\mathbf{x} = \|\mathbf{x}\|\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{A.48}$$

Alternatively, if vector $\mathbf{u}$ in Eq. (A.46) is chosen as

$$\mathbf{u} = \mathbf{x} + \|\mathbf{x}\|\mathbf{e}_1 \tag{A.49}$$

then

$$\mathbf{H}\mathbf{x} = -\|\mathbf{x}\|\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{A.50}$$