Optimization Homework Report

Ke-Cen Sha 2200010611

December 13, 2024

目录

1 Problem Description

Consider the following group Lasso optimization problem:

$$\min_{x\in\mathbb{R}^{n\times l}} \frac{1}{2}\|Ax-b\|_F^2 + \mu\|x\|_{1,2}, \tag{1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^{n\times l}$, $\mu > 0$ are given constants, and

$$\|x\|_{1,2} = \sum_{i=1}^{n} \|x(i,1:l)\|_2.$$

1. Solve the above optimization problem using CVX with Gurobi and MOSEK solvers, respectively.
2. First, write an equivalent model of (1) that can be directly solved by MOSEK and Gurobi, and implement the corresponding code.
3. Solve the optimization problem (1) using the following methods:
   (a) Subgradient method on the primal problem;
   (b) Smoothed gradient method on the primal problem;
   (c) Proximal point algorithm on the primal problem;
   (d) Accelerated proximal point algorithm;
   (e) Augmented Lagrangian method on the dual problem;
   (f) Alternating Direction Method of Multipliers (ADMM) on the dual problem;
   (g) ADMM with linearization technique applied to the primal problem.

2 Problem Analysis and Results

To ensure reproducibility, all results below are generated under a fixed random seed. Setting the seed to "24847563" at the beginning of the code will reproduce the reported results.

2.1 Problem 1

The original problem can be directly expressed in CVX. The results obtained using the two different solvers are shown below. Numerical values are reported with 5 significant digits[1], and time is reported in seconds with two decimal places[2]. From Table 1, we observe that the performance of the two solvers is quite similar.

solver	Fval	Time	Sparisity	Errfun_exact	Err-to-cvx-mosek	Err-to-cvx-gurobi
cvx-mosek	6.0715e-01	8.42	0.110	4.1480e-05	0	2.5553e-07
cvx-gurobi	6.0715e-01	9.96	0.110	4.1312e-05	2.5553e-07	0

表 1: Results using CVX solver

[1] All subsequent tables follow this standard.

[2] The initial solving times of 8–9 seconds were unnecessary; later optimization reduced the time (see final table). It was found that the initial model construction was not concise enough.

2.2 Problem 2

For the MOSEK solver, we reformulate the original problem as a second-order cone program (SOCP):

$$
\begin{array}{lll}
\min\limits_{x\in\mathbb{R}^{n\times l},t\in\mathbb{R},s\in\mathbb{R}^n} & \frac{t}{2}+\mu\cdot\mathbf{1}^\top s, & \\
\text{s.t.} & \|x(i,1:l)\|_2\le s(i), & 1\le i\le n \\
 & \|Ax-b\|_F^2\le t. &
\end{array} \tag{2}
$$

Here, $\mathbf{1}=(1,1,\cdots,1)^\top$. In practice, the final rotated quadratic cone constraint must be converted into a standard quadratic cone constraint. According to MOSEK syntax, we create $nl+n+1$ variables to represent $x(i,j)$, $s(i)$, and t. MOSEK requires all quadratic cone constraints to be in standard form:

$$
Fx+g\in\mathcal{S},
$$

where $\mathcal{S}=\{(u,v)^\top\in\mathbb{R}^{k+1}|\|u\|_2\le v\}$. We write the corresponding F,g for the $n+1$ cone constraints in (2), then combine them to solve the problem using MOSEK (see code for details).

For the Gurobi solver, we reformulate the problem as a quadratically constrained quadratic program (QCQP):

$$
\begin{array}{lll}
\min\limits_{u\in\mathbb{R}^{m\times l},x\in\mathbb{R}^{n\times l},s\in\mathbb{R}^n} & \frac{1}{2}\|u\|_F^2+\mu\cdot\mathbf{1}^\top s, & \\
\text{s.t.} & x(i,1:l)x(i,1:l)^\top-s(i)^2\le 0, & 1\le i\le n \\
 & s(i)\ge 0, & \\
 & Ax-u=-b. &
\end{array} \tag{3}
$$

Here, $\mathbf{1}=(1,1,\cdots,1)^\top$. Similarly, we create $(m+n)l+n$ variables to represent u, x, and s. According to Gurobi syntax, we need to specify the linear and quadratic coefficient matrices F, $Q\in\mathbb{R}^{[(m+n)l+n]\times[(m+n)l+n]}$, variable bounds lb, $ub\in\mathbb{R}^{(m+n)l+n}$, and the n quadratic constraints in (3). With these, we can solve the problem using Gurobi (see code for details).

Below are the results obtained by directly using MOSEK and Gurobi solvers after constructing the above models. From Table 2, we see that the Gurobi model performs worse than CVX-MOSEK, CVX-Gurobi, and MOSEK[3], especially in sparsity. Thus, there is room for improvement. Note that problem (3) has equivalent forms that can be implemented in Gurobi, such as:

$$
\begin{array}{lll}
\min\limits_{x\in\mathbb{R}^{n\times l},s\in\mathbb{R}^n,t\in\mathbb{R}} & \frac{t}{2}+\mu\cdot\mathbf{1}^\top s, & \\
\text{s.t.} & x(i,1:l)x(i,1:l)^\top-s(i)^2\le 0, & 1\le i\le n \\
 & s(i)\ge 0, & \\
 & \|Ax-b\|_F^2\le t. &
\end{array} \tag{4}
$$

or

$$
\begin{array}{lll}
\min\limits_{x\in\mathbb{R}^{n\times l},s\in\mathbb{R}^n} & \frac{1}{2}\sum\limits_{i=1}^{l}(x_i^\top A^\top Ax_i-2b_i^\top x_i+b_i^\top b_i)+\mu\cdot\mathbf{1}^\top s, & \\
\text{s.t.} & x(i,1:l)x(i,1:l)^\top-s(i)^2\le 0, & 1\le i\le n \\
 & s(i)\ge 0. &
\end{array} \tag{5}
$$

Here, x_i and b_i denote the i-th column of x and b, respectively. These alternative models perform worse than (3), even failing to converge to the optimal solution. Results are shown in Table 3. Therefore, when using Gurobi for QCQP, it is advisable to simplify the quadratic form in the objective (compare (3) and (4)) and prefer linear constraints over quadratic ones (compare (3) and (5)).

[3]Gurobi version affects results; Gurobi 11.0.3 gives slightly better results. This experiment uses Gurobi 12.0.0.

solver	Fval	Iter	Time	Sparisity	Errfun_exact	Err-to-cvx-mosek	Err-to-cvx-gurobi
mosek	6.0715e-01	11	0.79	0.110	4.1413e-05	1.0231e-07	1.6911e-07
gurobi	6.0715e-01	11	0.83	0.121	4.3251e-05	2.2605e-06	2.4753e-06

表 2: Results using MOSEK and Gurobi directly

solver	Fval	Iter	Time	Sparisity	Errfun_exact	Err-to-cvx-mosek	Err-to-cvx-gurobi
gurobi(3)	6.0715e-01	11	0.83	0.121	4.3251e-05	2.2605e-06	2.4753e-06
gurobi(4)	6.8843e-01	30	2.07	0.992	2.9837e-03	2.9617e-03	2.9618e-03
gurobi(5)	6.0759e-01	15	0.92	0.274	5.1274e-05	1.1977e-05	1.2164e-05

表 3: Results using different Gurobi models

2.3 Problem 3(a)

Let the objective function be $f(x,\mu)$. The subgradient of $f(x,\mu)$ with respect to x (row-wise) is:

$$\partial f(x,\mu)(i,:) = (A^\top(Ax-b))(i,:) + \begin{cases} \mu x(i,1:l)/\|x(i,1:l)\|_2 & \text{if } x(i,1:l) \neq 0, \\ \{\mu g \in \mathbb{R}^{1\times l} \| \|g\|_2 \leq 1\} & \text{else.} \end{cases}$$

We apply the subgradient method with BB step size, backtracking line search, and continuation strategy. The algorithm is as follows:

```
Algorithm 1: Subgradient method for solving problem (1)
Input: Initial point $x_0$, $A$, $b$, $\mu$, constant $L$, continuation parameter $\rho > 1$, $k \in \mathbb{N}^*$, $iter = 0$, $f^* =$ inf.
Result: Return optimal solution $x_0$, minimum value $f^*$, number of iterations $iter$.
1  $v = \mu \times \rho^k$; $x_1 = x_0 - \frac{1}{L}\partial f(x_0, v)$;
2  while $v \neq \mu \vee |f(x_1,\mu) - f(x_0,\mu)| \geq 10^{-10}$ do
3      $s = x_1 - x_0$, $y = \partial f(x_1, v) - \partial f(x_0, v)$, $g = \partial f(x_1, v)$;
4      $\alpha_{BB} = \|s\|_F^2/\langle s, y\rangle$, $a = \max(\alpha_{BB}, L)$;
5      while $a > \frac{1}{L} \wedge f(x_1,v) - f(x_1 - ag, v) < \frac{1}{2L}\|g\|_F^2$ do
6          $a = a/2$;
7      end
8      while $f(x_1,v) - f(x_1 - ag, v) < 0$ do
9          $a = a/2$;
10         if $a < 10^{-15}$ then
11             break
12         end
13     end
14     $x_0 = x_1$, $x_1 = x_1 - ag$, $iter = iter + 1$, $f^* = \min(f^*, f(x_0,\mu))$;
15     if $v > \mu \wedge \|f(x_0,v) - f(x_1,v)\| \leq 10^{-5} \cdot v$ then
16         $v = v/\rho$;
17     end
18 end
Output: $x_0$, $iter$, $f^*$
```

In this algorithm, we first increase μ to v, solve the problem for $f(x,v)$, and gradually decrease v until $v = \mu$. Theoretically, non-differentiable functions lack Lipschitz continuity; here, to prevent BB step size from being too large, we set $L = \|A^\top A\|_2$[4]. To prevent BB step size from being too small, we set a minimum step size of 10^{-15}.

[4] L is inspired by gradient descent; when $\|x\|$ is not too small, f can be treated as having L-Lipschitz continuous gradient

Additionally, subgradient selection is important. When $\|x(i,:)\|_2$ is very small, the error in $x(i,1:l)/\|x(i,1:l)\|_2$ becomes large. Therefore, a threshold ϵ is set: when $\|x(i,:)\|_2 \leq \epsilon$, treat it as zero[5], and set the subgradient component to $\partial f(x,\mu)(i,:) = (A^\top(Ax-b))(i,:) + x(i,:)/\epsilon$. Experimentally, $\epsilon = 10^{-6}$ yields good results.

Recommended continuation parameters: $\rho = 5$, $k = 5$. Backtracking line search is used in two stages: when $a > L$, use a weakened Armijo condition ($c_1 = 1/2$, requiring descent from $a/2$ to $1/(2L)$); when $a < L$, ensure monotonic decrease.

Below are the results of the subgradient method. For comparison, results without BB step size or continuation are also shown. In this case, omitting BB still performs well, but BB’s importance will be evident in Problem 3(b). Without continuation, the algorithm requires about 200,000 iterations to converge.

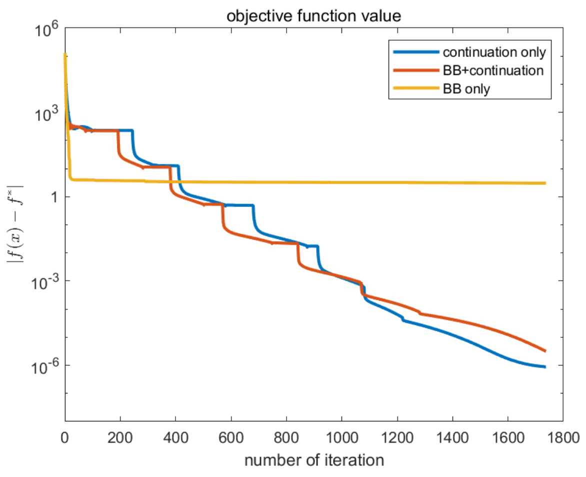


图 1: Results obtained by subgradient method

2.4 Problem 3(b)

We use a smoothing strategy to accelerate SGD, applying the Huber smoothing function to $\|x(i,:)\|_2$. The Huber function is:

$$h_u(x) = \begin{cases} \frac{x^2}{2u} & \textbf{if } 0 \leq x \leq u, \\ x - \frac{u}{2} & \textbf{if } x \geq u. \end{cases}$$

The smoothed function is:

$$f_u(x,\mu) = \frac{1}{2}\|Ax-b\|_F^2 + \mu \sum_{i=1}^{n} h_u(\|x(i,:)\|_2),$$

and satisfies

$$|f_u(x,\mu) - f(x,\mu)| \leq \frac{\mu n u}{2}.$$

It can be shown that $h_u(x)$ has a $\frac{1}{u}$-Lipschitz continuous gradient, so $f_u(x,\mu)$ is $(\|A^\top A\|_2 + \frac{\mu\sqrt{n}}{u})$-Lipschitz continuous[6]. Thus, standard gradient methods, or BB step size with line search, can be used. As in 3(a), we use continuation. A small trick: as u decreases, the Lipschitz constant increases, so we simultaneously decrease μ with u to keep the Lipschitz constant small in early iterations for faster convergence.

Below are the results of the smoothed gradient descent. Four experiments were conducted. From Figure 2, BB step size plays a key role in accelerating convergence, more so than continuation. Combining both yields the fastest convergence.

[5]Strictly speaking, do not use $x(i,1:l)/\|x(i,1:l)\|_2$, but choose any g with $\|g\|_2 \leq 1$.

[6]In practice, $L = \|A^\top A\|_2 + \frac{\mu}{u}$ works well

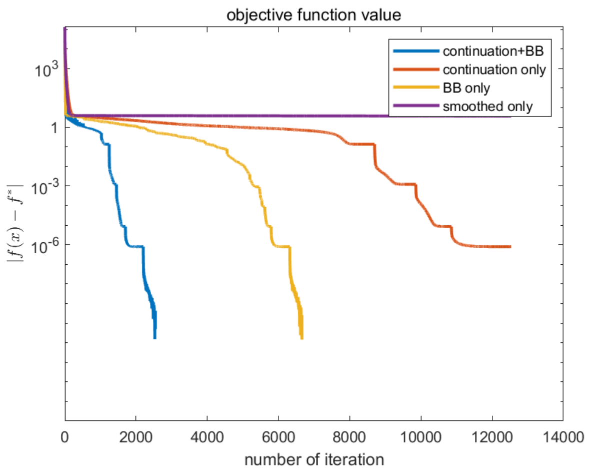


图 2: Results obtained by smoothed gradient method

Although the smoothed method requires slightly more iterations, it achieves better accuracy. The non-smooth subgradient method is highly sensitive to parameters and requires many iterations for high precision. Table 4 compares the two.

solver	Fval	Iter	Time	Sparisity	Errfun_exact	Err-to-cvx-mosek	Err-to-cvx-gurobi
non-smoothed	6.0715e-01	1983	0.51	0.113	4.5045e-05	4.0763e-06	4.2597e-06
smoothed	6.0715e-01	3325	0.70	0.110	4.1520e-05	1.9296e-07	2.3676e-07

表 4: Comparison of results from 3(a) and 3(b)

2.5 Problem 3(c), 3(d)

Let $f(x) = \frac{1}{2}\|Ax-b\|_F^2$, $h_\mu(x) = \mu\|x\|_{1,2}$. The proximal operator of $h_\mu(x)$ (row-wise) is:

$$\mathrm{prox}_{th_\mu}(y)(i,:) = \begin{cases} 0 & \text{if } \|y(i:)\|_2 \le t\mu, \\ y(i,:) - t\mu\frac{y(i,:)}{\|y(i,:)\|_2} & \text{else.} \end{cases} \quad ,1 \le i \le n$$

Let $L = \|A^\top A\|_2$. Then $f(x)$ has L-Lipschitz continuous gradient, so the standard proximal gradient method applies:

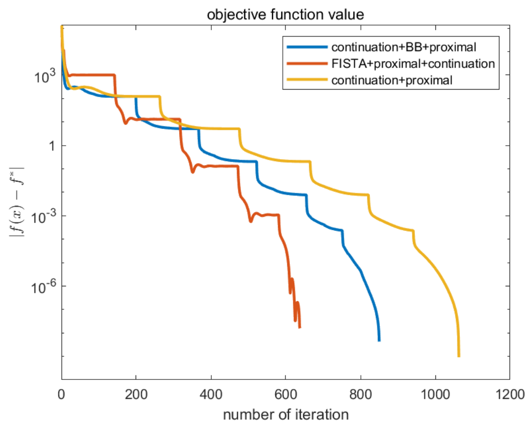
$$y_k = x_k - \frac{1}{L}\nabla f(x_k);$$
$$x_{k+1} = \mathrm{prox}_{\frac{1}{L}h_\mu}(y_k).$$

We try BB step size and continuation for acceleration, similar to Algorithm 1. For 3(d), we use FISTA acceleration and continuation, as shown in Algorithm 2:

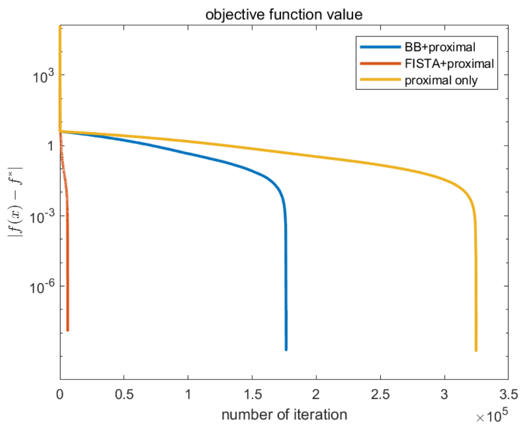
Algorithm 2: FISTA-accelerated proximal gradient method for solving the optimization problem

```
Input: Matrix $A \in \mathbb{R}^{m\times n}$, vector $b \in \mathbb{R}^l$, parameter $\mu$, initial point $x_0$, constant $L$, continuation parameter
       $\rho > 1$, $N \in \mathbb{N}^*$.
Result: Return optimal solution $x_0$, minimum value $f^*$, number of iterations $iter$.
1  $iter = 0$; $k = 1$; $v = \mu \times \rho^N$; $x_1 = \text{prox}_{\frac{1}{L}h_v}\left(x_0 - \frac{1}{L}A^T(Ax_0 - b), \frac{1}{L}\right)$;
2  while $v \geq \mu$ do
3      $f^* = f(x_0)$; $iter = iter + 1$; $k = k + 1$;
4      $y = x_1 + \frac{k-2}{k+1}(x_1 - x_0)$; $x_0 = x_1$; $x_1 = \text{prox}_{\frac{1}{L}h_v}\left(y - \frac{1}{L}A^T(Ay - b), \frac{1}{L}\right)$;
5      if $\|x_0 - x_1\|_F < 10^{-5} \wedge v > \mu$ then
6          $v = v/\rho$; $k = 1$;
7      end
8      if $v == \mu \wedge \|x_0 - x_1\|_F < 10^{-8}$ then
9          break;
10     end
11 end
Output: $x_0$, $iter$, $f^*$
```

Figure 3 shows the numerical results using the proximal gradient method, with and without continuation. Recommended parameters: $\rho = 10, N = 5$. Unlike smoothing, continuation has a more significant effect than BB in the proximal method. Compared to subgradient and smoothed gradient methods, the proximal method converges faster and achieves higher accuracy. Table 5 shows performance metrics (all with continuation).



(a) Proximal algorithm with continuation



(b) Proximal algorithm without continuation

图 3: Results obtained by proximal algorithm

solver	Fval	Iter	Time	Sparisity	Errfun_exact	Err-to-cvx-mosek	Err-to-cvx-gurobi
proximal+BB	6.0715e-01	947	0.13	0.110	4.1329e-05	2.4914e-07	2.6723e-07
proximal+FISTA	6.0715e-01	841	0.04	0.110	4.1328e-05	2.4999e-07	2.5531e-08

表 5: Comparison of results from 3(d) and 3(e)

2.6 Problem 3(e)

The dual problem of the original is:

$$\min_{y\in\mathbb{R}^{m\times l}} \quad \frac{1}{2}\|y\|_F^2 - \langle y, b\rangle,$$
$$\text{s.t.} \quad \|A^\top y\|_{1,2} \leq \mu.$$

To make the constraint differentiable, we reformulate it as:

$$\begin{aligned} \min_{y\in\mathbb{R}^{m\times l}, s\in\mathbb{R}^{n\times l}} \quad & \frac{1}{2}\|y\|_F^2 - \langle y, b\rangle, \\ \text{s.t.} \quad & \|(A^\top y)_i\|^2 \le \mu^2. \qquad 1 \le i \le n \end{aligned}$$

where $(A^\top y)_i$ denotes the i-th row of $A^\top y$. Using elimination, the augmented Lagrangian is:

$$L_\sigma(y,\lambda) = \frac{1}{2}\|y\|_F^2 - \langle y, b\rangle + \frac{\sigma}{2}\sum_{i=1}^{n}(\max\{\frac{\lambda_i}{\sigma} + \|(A^\top y)_i\|_2^2 - \mu^2, 0\}^2 - \frac{\lambda_i^2}{\sigma^2}).$$

Define multiplier update:

$$U_i(\sigma, y, \lambda) = \max\{\lambda_i + \sigma(\|(A^\top y)_i\|_2^2 - \mu^2), 0\}, \quad 1 \le i \le n$$

$$\vec{U}(\sigma, y, \lambda) = (U_i(\sigma, y, \lambda))^\top.$$

Gradient of L_σ w.r.t. y:

$$\nabla_y L_\sigma(y,\lambda) = y - b + 2A \cdot \text{diag}(\vec{U}(\sigma, y, \lambda)) \cdot A^\top y.$$

Multiplier update:

$$\lambda_i^{k+1} = U_i(\sigma_k, y_{k+1}, \lambda^k) = \max\{\lambda_i^k + \sigma_k(\|(A^\top y_{k+1})_i\|_2^2 - \mu^2), 0\}. \quad 1 \le i \le n$$

Constraint violation:

$$V(\sigma_k, y_{k+1}, \lambda^k) = \frac{\|\vec{U}(\sigma_k, y_{k+1}, \lambda^k) - \lambda^k\|_2}{\sigma_k}.$$

The augmented Lagrangian method is shown in Algorithm 3:

Algorithm 3: Augmented Lagrangian method for dual problem

```
Input: Matrix $A$, vector $b$, parameter $\mu$, penalty growth rate $\rho$, initial penalty $\sigma$, upper bound $M$, target
       accuracy $\eta$, target violation $\varepsilon$, initial accuracy $\eta_0$, initial violation $\varepsilon_0$, decay exponent $\beta$
1  Initialization $\lambda^0 = \mathbf{0}_{n\times 1}$, $y_0 = \mathbf{0}_{m\times l}$, outer loop $k = 0$
2  while $V(\sigma_k, y_k, \lambda^k) > \varepsilon \vee \|\nabla L_{\sigma_k}(y_k, \lambda^k)\|_F > \eta$ do
3      if $\sigma > M$ then
4          break
5      end
6      Solve subproblem $y_{k+1} = \text{argmin}_z L_\sigma(z, \lambda^k)$ using Nesterov acceleration from $y_k$, ensuring
        $\|\nabla_y L_\sigma(y_{k+1}, \lambda)\| \le \eta_k$
7      if $V(\sigma_k, y_k, \lambda^k) \le \varepsilon_k$ then
8          if $V(\sigma_k, y_k, \lambda^k) \le \varepsilon_0 \wedge \|\nabla_y L_{\sigma_k}(y_k, \lambda^k)\|_F \le \eta_0$ then
9              break
10         end
11         $\lambda^{k+1} = \vec{U}(\sigma_k, y_{k+1}, \lambda^k)$
12         $\varepsilon_{k+1} = \varepsilon_{k+1}/\rho$
13         $\sigma_{k+1} = \sigma_k$
14         $\eta_{k+1} = \eta_k$
15     else
16         $\sigma_{k+1} = \rho \cdot \sigma_k$
17         $\eta_{k+1} = \eta_k/\rho^\beta$
18         $\lambda^{k+1} = \lambda^k$
19         $\varepsilon_{k+1} = \varepsilon_{k+1}$
20     end
21     $k = k + 1$
22 end
Result: Minimizer $y^*$, multiplier $\lambda_*$, outer iterations $k$
```

The Nesterov method is similar to Algorithm 2 (FISTA) without proximal step. Recommended parameters: $\sigma_0 = 10^5$, $\rho = 10$, $\beta = \lg 5$, $M = 10^7$, $\varepsilon = 10^{-7}$, $\eta = 10^{-5}$, $\eta_0 = \varepsilon_0 = 10^{-2}$. A penalty upper bound is set. Empirically, initial penalty should be large; accuracy is harder to satisfy than violation, so accuracy decreases slower (controlled by β). Numerical results are in the final table.

2.7 Problem 3(f)

The dual can also be written as:

$$\begin{aligned} \min_{y\in\mathbb{R}^{m\times l}, s\in\mathbb{R}^{n\times l}} \quad & \frac{1}{2}\|y\|_F^2 - \langle y, b\rangle, \\ \text{s.t.} \quad & A^\top y - s = 0, \\ & \|s\|_{1,2} \le \mu. \end{aligned}$$

Let $\mathbf{C} = \{s \in \mathbb{R}^{n\times l} | \|s\|_{1,2} \le \mu\}$, then:

$$\begin{aligned} \min_{y\in\mathbb{R}^{m\times l}, s\in\mathbb{R}^{n\times l}} \quad & \frac{1}{2}\|y\|_F^2 - \langle y, b\rangle + \mathbf{1}_{\mathbf{C}}(s), \\ \text{s.t.} \quad & A^\top y - s = 0. \end{aligned}$$

This form allows ADMM. The augmented Lagrangian is:

$$L_\sigma(y, s, \lambda) = \frac{1}{2}\|y\|_F^2 - \langle y, b\rangle + \mathbf{1}_{\mathbf{C}}(s) + \langle \lambda, A^\top y - s\rangle + \frac{\sigma}{2}\|A^\top y - s\|_F^2.$$

Fix s, minimize over y: quadratic, solution is:

$$y^* = (I + \sigma AA^\top)^{-1}(b + A\lambda + \sigma As). \tag{6}$$

Fix y, minimize over s: has explicit solution[7]:

$$s_i = \left((A^\top y)_i + \frac{\lambda_i}{\sigma}\right) \min\{\frac{\mu}{\|(A^\top y)_i + \frac{\lambda_i}{\sigma}\|}, 1\}, \quad 1 \le i \le n \tag{7}$$

where s_i is the i-th row of s. In inner loop, compute Cholesky of $I + \sigma AA^\top$, then solve. The ADMM algorithm with dynamic penalty adjustment based on primal/dual feasibility is:

Algorithm 4: ADMM for dual problem

```
Input: Matrix $A$, vector $b$, parameter $\mu$, penalty rate $\rho$, initial $\sigma$, coordination rate $\eta$, tolerance $\varepsilon$
1  Initialize $\lambda^0 = \mathbf{0}_{n\times 1}$, $y_0 = \mathbf{0}_{m\times l}$, $s^1 = \mathbf{1}_{n\times l}$, $s^0 = \mathbf{0}_{n\times l}$, $k = 0$, Cholesky $I + \sigma AA^\top = R^\top R$
2  while $\|A(s^1 - s^0)\|_F > \varepsilon$ do
3      $s^0 = s^1$
4      $y_0 = R^{-1}(R^{-\top}(b + A\lambda^k + \sigma As^1))$
5      $s_i^1 = \left((A^\top y_0)_i + \frac{\lambda_i^k}{\sigma}\right) \min\left\{\frac{\mu}{\|(A^\top y_0)_i + \frac{\lambda_i^k}{\sigma}\|}, 1\right\}, \quad 1 \le i \le n$
6      $\lambda^{k+1} = \lambda^k + \sigma(A^\top y_0 - s^1)$
7      if $\|s^1 - A^\top y_0\|_F > \eta\|A(s^1 - s^0)\|_F$ then Enlarge penalty
8          $\sigma = \rho\sigma$, $R = \text{chol}(I + \sigma AA^\top)$
9      else if $\|A(s^1 - s^0)\|_F > \eta\|s^1 - A^\top y_0\|_F$ then Reduce penalty
10         $\sigma = \sigma/\rho$, $R = \text{chol}(I + \sigma AA^\top)$
11     end
12     $k = k + 1$
13 end
Result: Minimizer $y_0$, multiplier $\lambda^k$, iterations $k$
```

Recommended: $\rho = 2$, $\sigma = 1$, $\eta = 10$, $\varepsilon = 10^{-11}$. Since each subproblem is solved exactly, parameter choice is flexible.

[7]In code, no need to write per component.

2.8 Problem 3(g)

Rewrite the original problem:

$$\begin{aligned} \min_{s\in\mathbb{R}^{m\times l}, x\in\mathbb{R}^{n\times l}} \quad & \frac{1}{2}\|s\|_F^2 + \mu\|x\|_{1,2} \\ \text{s.t.} \quad & Ax - s - b = 0. \end{aligned}$$

The augmented Lagrangian is:

$$L_\sigma(s,x,\lambda) = \frac{1}{2}\|s\|_F^2 + \mu\|x\|_{1,2} + \langle \lambda, Ax - s - b\rangle + \frac{\sigma}{2}\|Ax - s - b\|_F^2.$$

Fix x, minimize over s: quadratic, solution:

$$s^* = \frac{\sigma(Ax-b)+\lambda}{1+\sigma}.$$

Fix s, use linearization: instead of solving

$$x^* = \arg\min_y \mu\|y\|_{1,2} + \frac{\sigma}{2}\|Ay - s - b + \frac{\lambda}{\sigma}\|_F^2,$$

which has no closed form, we minimize its quadratic approximation at current point (s_1, x_1, λ_1):

$$x^* = \arg\min_y \mu\|y\|_{1,2} + \sigma y^\top A^\top (Ax_1 - s_1 - b + \frac{\lambda_1}{\sigma}) + \frac{1}{2\eta}\|y - x_1\|_F^2,$$

where η is step size. This is equivalent to a proximal gradient step:

$$x^* = \text{prox}_{\eta\mu\|\cdot\|_{1,2}}\left(x_1 - \eta\sigma A^\top (Ax_1 - s_1 - b + \frac{\lambda_1}{\sigma})\right). \tag{8}$$

Step size uses BB rule (with Armijo + backtracking): $\eta = \frac{\|x_1 - x_0\|_F^2}{\sigma\|A(x_1-x_0)\|_F^2}$, where x_0 is previous iterate. Final algorithm:

Algorithm 5: ADMM with linearization for primal problem

```
Input: Matrix $A$, vector $b$, parameter $\mu$, penalty rate $\rho$, initial $\sigma$, coordination $u$, tolerance $\varepsilon$
1  Initialize $\lambda = \mathbf{0}_{n\times 1}$, $s_0 = \mathbf{0}_{m\times l}$, $x_1 = \mathbf{1}_{n\times l}$, $x_0 = \mathbf{0}_{n\times l}$, $k = 0$
2  while $\|A(x_0 - x_1)\|_F > \varepsilon$ do
3      $r = \frac{\|x_0 - x_1\|_F^2}{\sigma\|A(x_0-x_1)\|_F^2}$
4      $x_0 = x_1, k = k+1$
5      $s_0 = (\sigma(Ax_1 - b) + \lambda)/(1+\sigma)$
6      $r = \text{linesearch}(r)$
7      $g = x_1 - r\sigma(A^\top(Ax_1 - b - s_0 + \lambda/\sigma))$
8      $x_1 = \text{prox}_{\eta\mu\|\cdot\|_{1,2}}(x_1 - \eta\sigma A^\top(Ax_1 - s_0 - b + \frac{\lambda_1}{\sigma}))$, $\lambda = \lambda + \sigma(Ax_1 - b - s_0)$
9      if $u\|A(x_0 - x_1)\|_F < \|Ax_1 - b - s_0\|_F$ then
10         $\sigma = \sigma \cdot \rho$
11     end
12     else if $u\|Ax_1 - b - s_0\|_F < \|A(x_0 - x_1)\|_F$ then
13         $\sigma = \sigma/\rho$
14     end
15 end
Result: Minimizer $x_1$, multiplier $\lambda$, iterations $k$
```

For clarity, line search details are omitted in Algorithm 5; see code for implementation. Recommended: $\rho = 10$, $u = 10$, $\varepsilon = 10^{-10}$, $\sigma = 10^{-4}$. Note σ should be small (ideally ≤ 1), as large σ weakens the proximal effect in (8).

3 Summary of Results

3.1 Overall Table

Below are performance metrics for all methods[8]. We adjusted accuracy so that solutions differ from CVX-MOSEK and CVX-Gurobi by less than 2.5×10^{-6} (except SGD, which cannot achieve this). Final CVX results were slightly corrected[9], so they differ slightly from earlier reports.

Method	CPU (s)	Iterations	Sparsity	Error to Exact	Error to CVX-Mosek	Error to CVX-Gurobi
CVX-Mosek	1.33	-1	0.110	4.15×10^{-5}	0.00	2.71×10^{-7}
CVX-Gurobi	1.10	-1	0.110	4.17×10^{-5}	2.71×10^{-7}	0.00
Mosek	0.48	11	0.110	4.14×10^{-5}	1.02×10^{-7}	2.92×10^{-7}
Gurobi	0.65	11	0.121	4.33×10^{-5}	2.26×10^{-6}	2.21×10^{-6}
SGD Primal	0.54	1983	0.113	4.50×10^{-5}	4.08×10^{-6}	3.96×10^{-6}
GD Primal	0.81	3325	0.110	4.15×10^{-5}	1.93×10^{-7}	1.86×10^{-7}
FGD Primal	0.56	6897	0.110	4.13×10^{-5}	2.52×10^{-7}	3.42×10^{-7}
ProxGD Primal	0.15	947	0.110	4.13×10^{-5}	2.49×10^{-7}	3.39×10^{-7}
FProxGD Primal	0.05	810	0.110	4.13×10^{-5}	2.50×10^{-7}	3.41×10^{-7}
ALM Dual	0.83	7608	0.110	4.14×10^{-5}	8.74×10^{-7}	8.63×10^{-7}
ADMM Dual	0.11	497	0.110	4.13×10^{-5}	2.57×10^{-7}	3.49×10^{-7}
ADMM Primal	0.20	539	0.110	4.13×10^{-5}	2.57×10^{-7}	3.49×10^{-7}

表 6: Performance comparison of different optimization algorithms

3.2 Visualization of Results

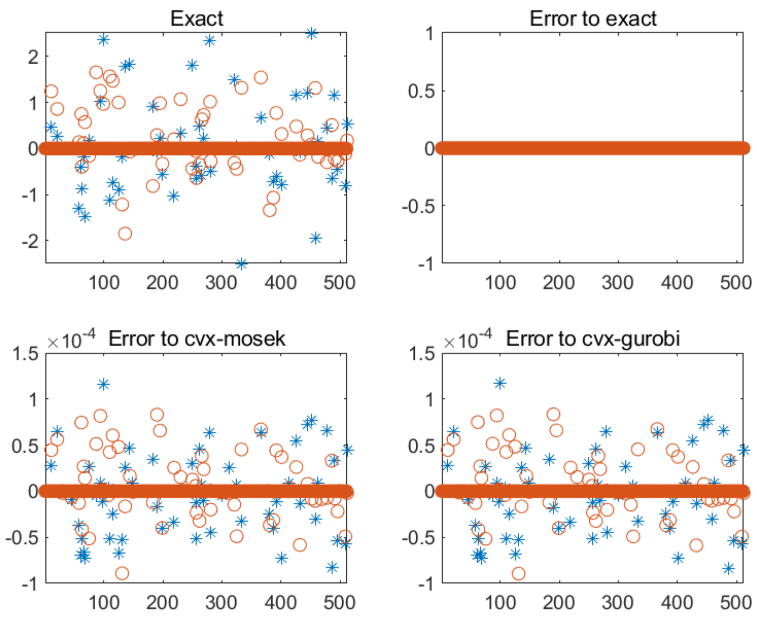


图 4: Exact solution and error analysis.

[8]Optimal value is 0.607146. -1 means iteration count not available. ALM Dual counts total inner iterations.

[9]Simplified model expression reduced CVX model-building time

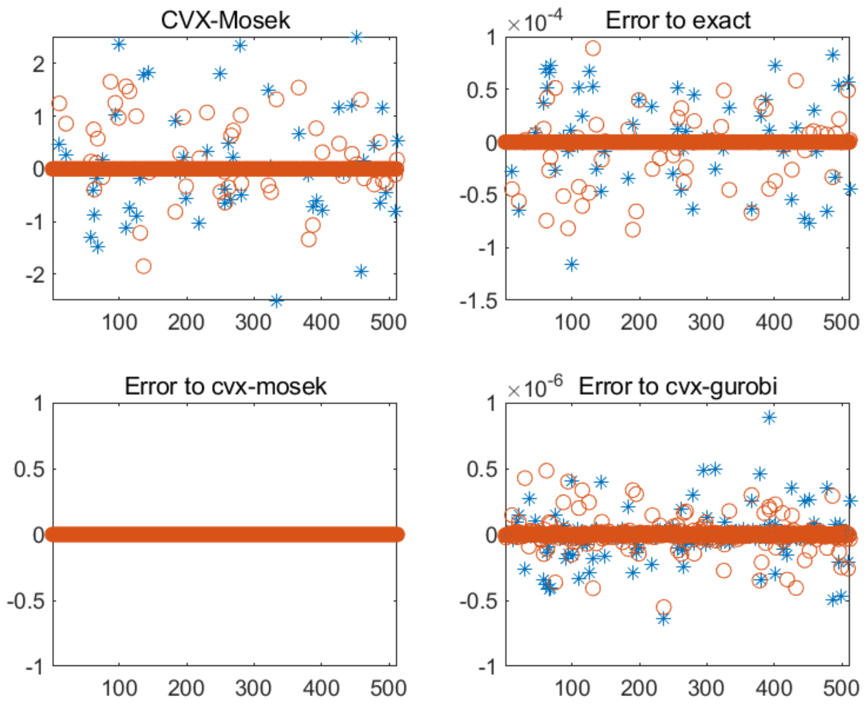


图 5: Error analysis for CVX-Mosek.

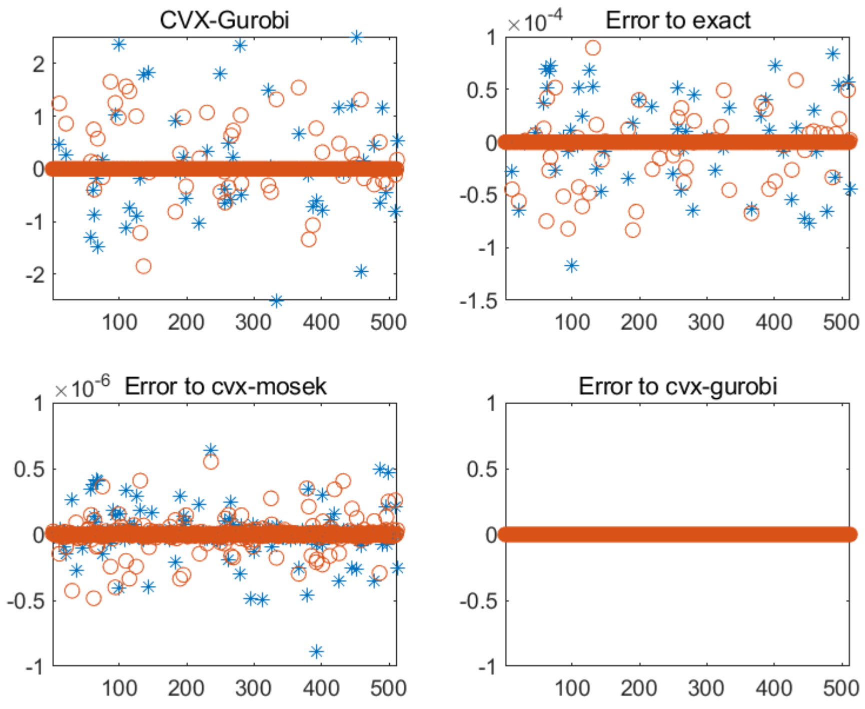


图 6: Error analysis for CVX-Gurobi.

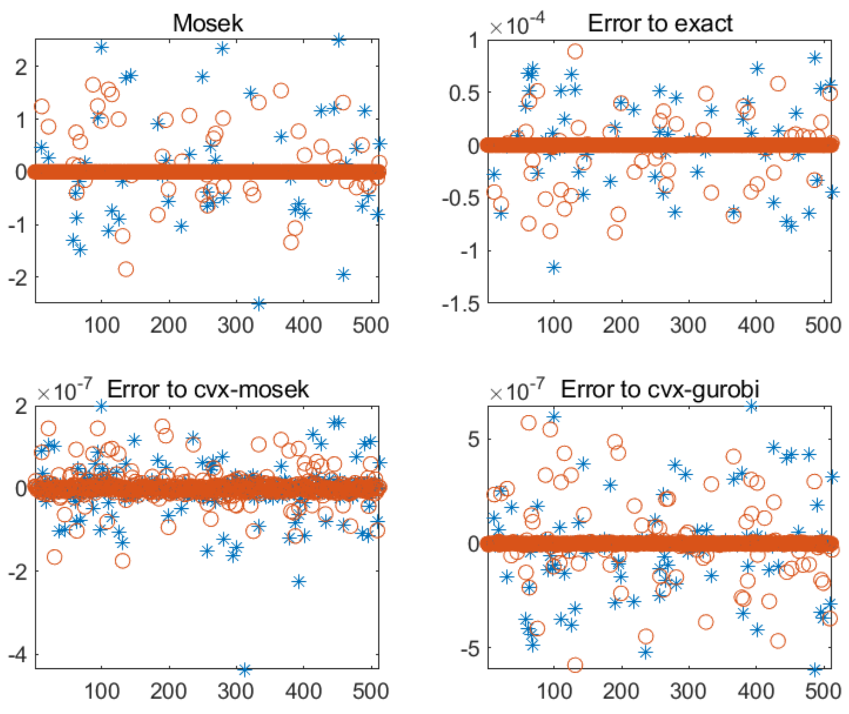


图 7: Error analysis for Mosek.

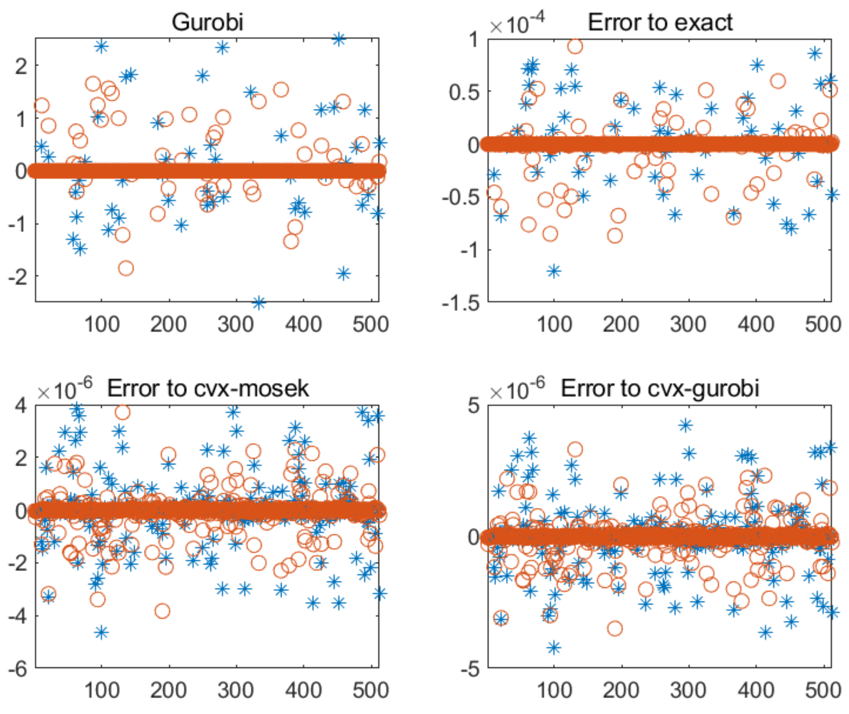


图 8: Error analysis for Gurobi.

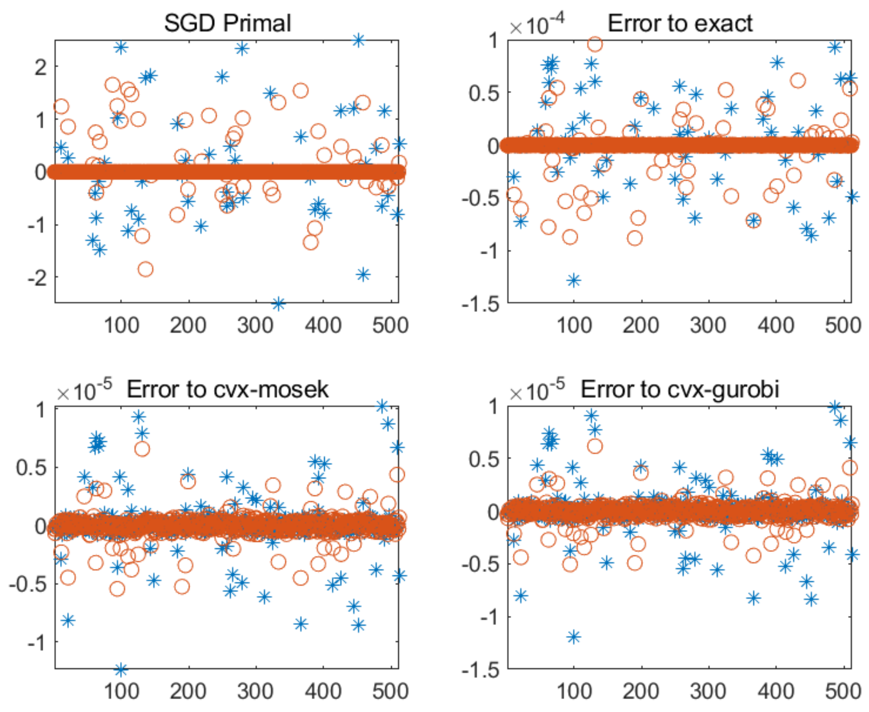


图 9: Error analysis for SGD Primal.

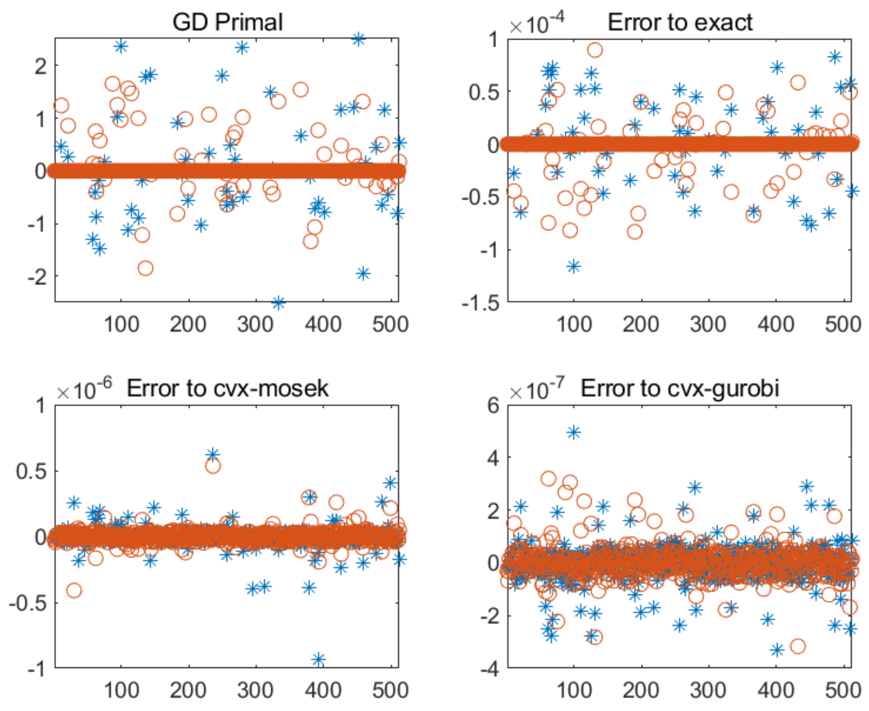


图 10: Error analysis for GD Primal.

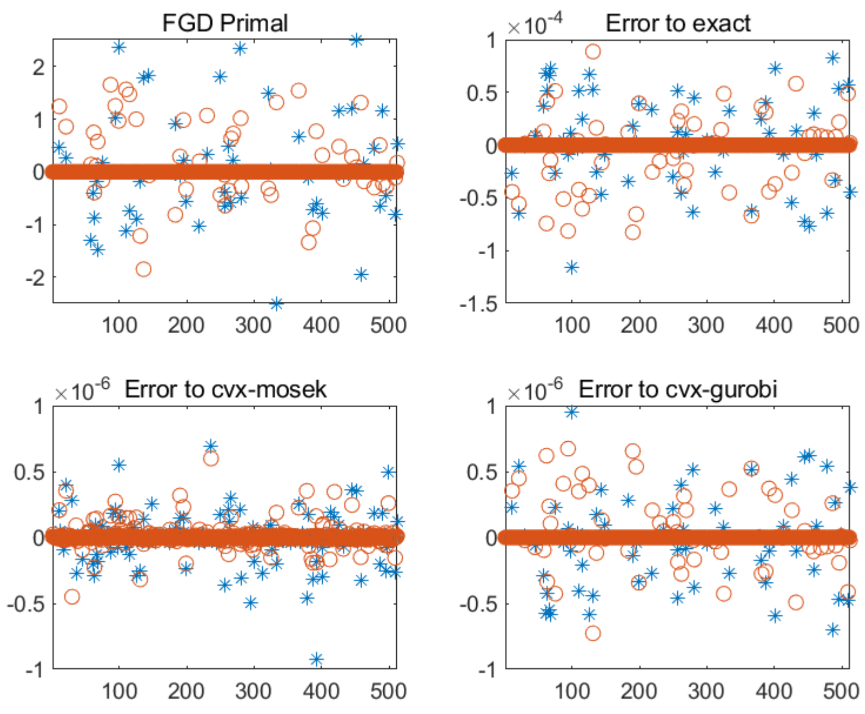


图 11: Error analysis for FGD Primal.

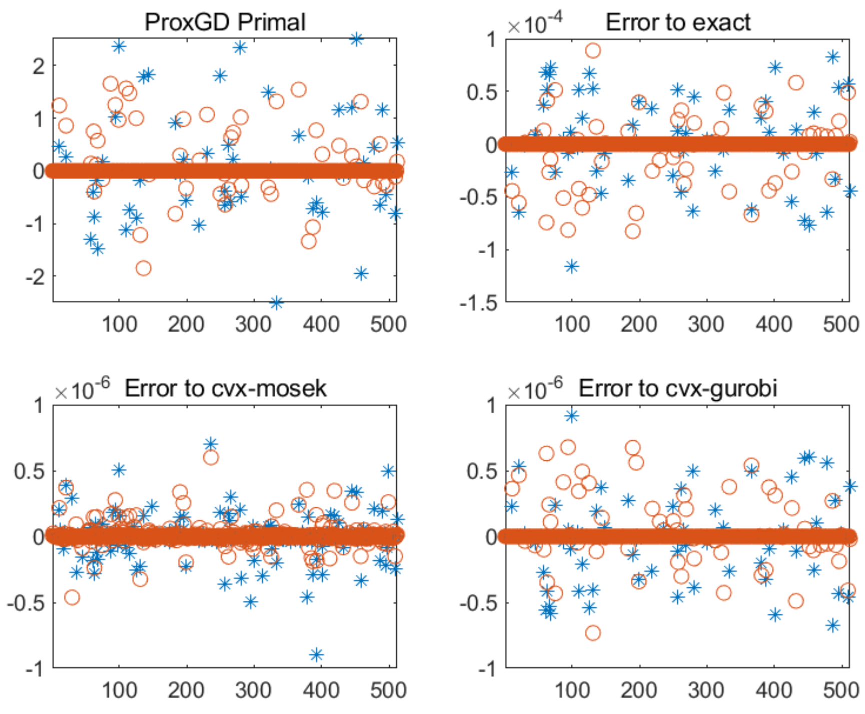


图 12: Error analysis for ProxGD Primal.

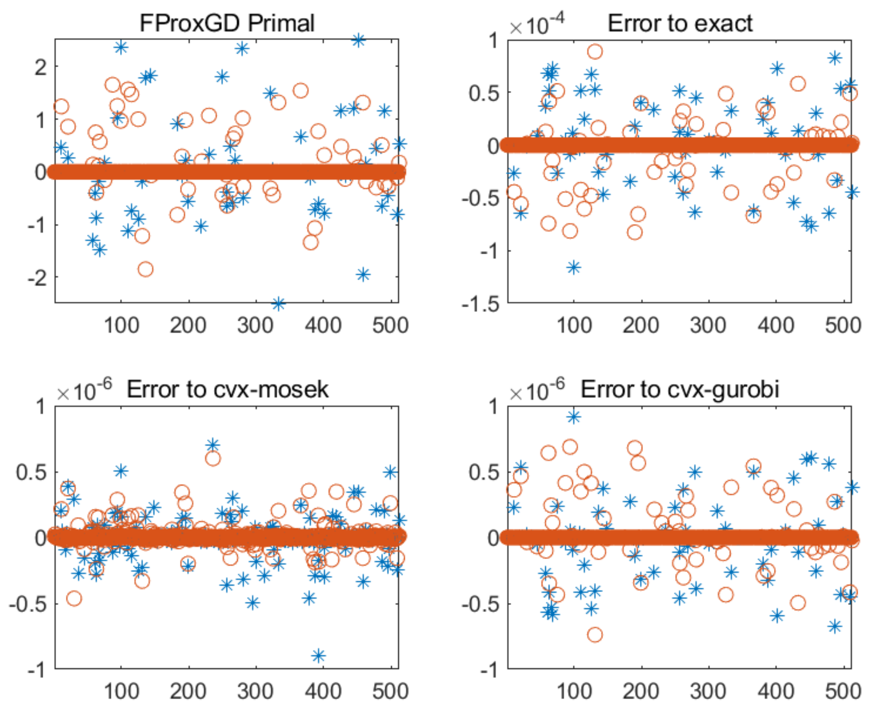


图 13: Error analysis for FProxGD Primal.

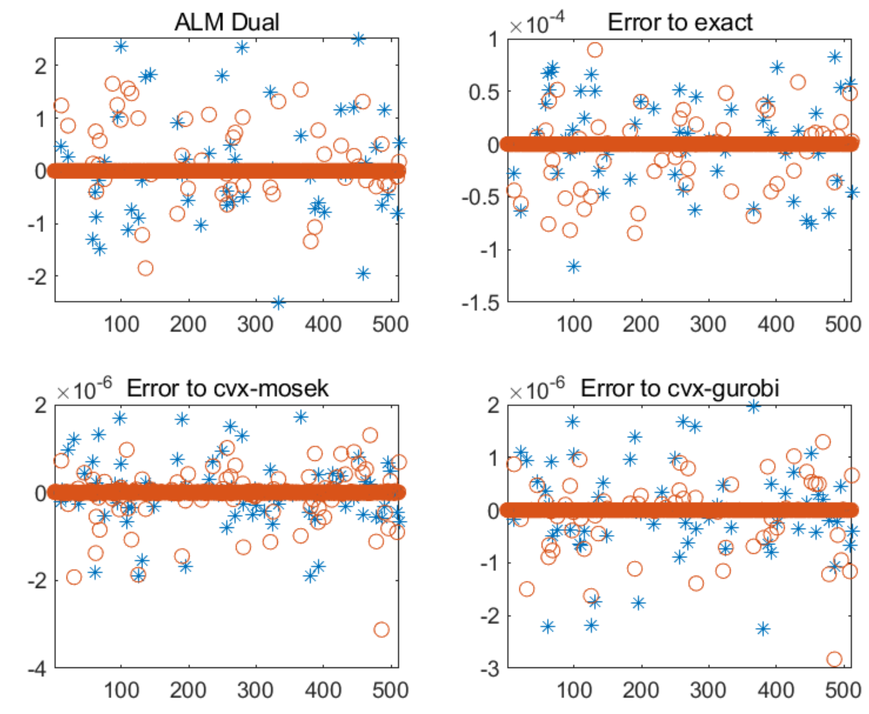


图 14: Error analysis for ALM Dual.

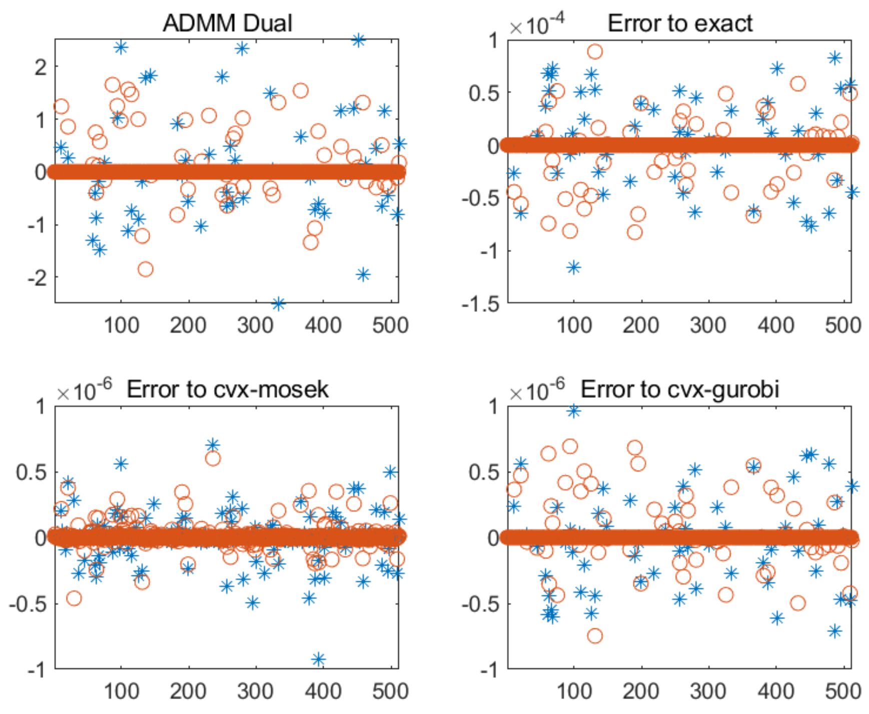


图 15: Error analysis for ADMM Dual.

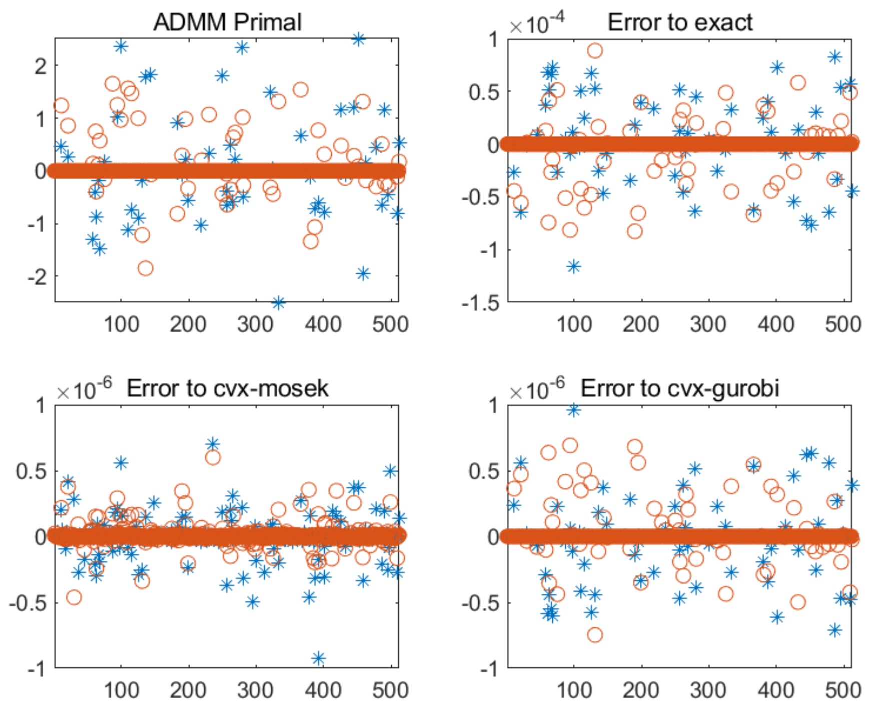


图 16: Error analysis for ADMM Primal.