Project 1 Report for Algorithms for Big-Data Analysis

Haiwen Huang 1500010657

April 16, 2018

1 Code

Together with this report, I submitted two folds of codes, which contains solutions to each of the problem.

1.1 l_1 minimization code

In the l_1 minimization fold, the main test file is $test1.m$, where you can test all the methods that solves the BP problem. I use six methods to solve the problem, and each method is contained in the corresponding function file which you can file in $test1.m$ file.

Moreover, I also provide a file for comparing between proximal gradient method and FISTA. It's named $comparison.m$.

1.2 Sparse Inverse code

In the sparse inverse fold, the main test file is $test2.m$, where you can test both cvx and first-order methods using data generated from Model1 and Model2. The cvx method is in $sparse_inverse.m$, and the first-order method is in $Nesterov.m$.

2 Algorithms for l_1 minimization

In this section we consider Basis Pursuit problem:

$$\min_x ||x||_1, \ s.t. \ Ax = b. \tag{1}$$

In total, we use 6 ways to solve this problem, and we will show the algorithms, numerical results and their interpretation below. The 6 methods are:

- CVX
- Mosek in CVX
- directly using Mosek

- Augmented Lagrangian method, where each augmented Lagrangian function is minimized using proximal gradient method
- Augmented Lagrangian method, where each augmented Lagrangian function is minimized using accelerated proximal gradient method(FISTA)
- Alternating direction method of multipliers(ADMM) for the dual problem

We will first give a summary of the numerical results of these methods. Then, since the first three methods are basically implementation, I will focus on the details of the later three methods. I will explain the algorithm, or derive the formulation for the dual problem. And interpret the results in more detail.

2.1 Summary of numerical results

We use two figures(Figure1 and Figure2) to illustrate both the qualitative and quantitative results.

2.2 Mosek Formula Derivation

In this section and next two sections, we will derive the formula for mosek, augmented lagrangian methods and ADMM.

The original problem (1) is equivalent to

$$\min_x \sum_i |x_i|,\ s.t.\ Ax = b. \tag{2}$$

Let $x_i = x_i^+ - x_i^-$, we transform the problem (2) into a linear programming problem:

$$\min_x \sum_i x_i^+ + x_i^-,\ s.t.\ Ax^+ - Ax^- = b\ x^+ \succeq 0,\ x^- \succeq 0 \tag{3}$$

Let new variable $s = \begin{bmatrix} x^+ \\ x^- \end{bmatrix}$, we can rewrite the problem as:

$$\min_s \mathbf{1}^T s,\ s.t.\ \begin{bmatrix} A & -A \end{bmatrix} s = b, \quad s \succeq 0; \tag{4}$$

This is the standard form of LP problem, and can be solved using mosek directly.p

2.3 Augmented Lagrangian Methods

Remember that our problem is $\min_x ||x||_1,\ s.t.\ Ax = b$. Therefore, we have Lagrangian function

$$L(x, y) = ||x||_1 + y^T(Ax - b)$$

, then we add the augmented term, get

$$L_\mu(x, y) = ||x||_1 + y^T(Ax - b) + \frac{1}{2\mu}||Ax - b||^2 \tag{5}$$

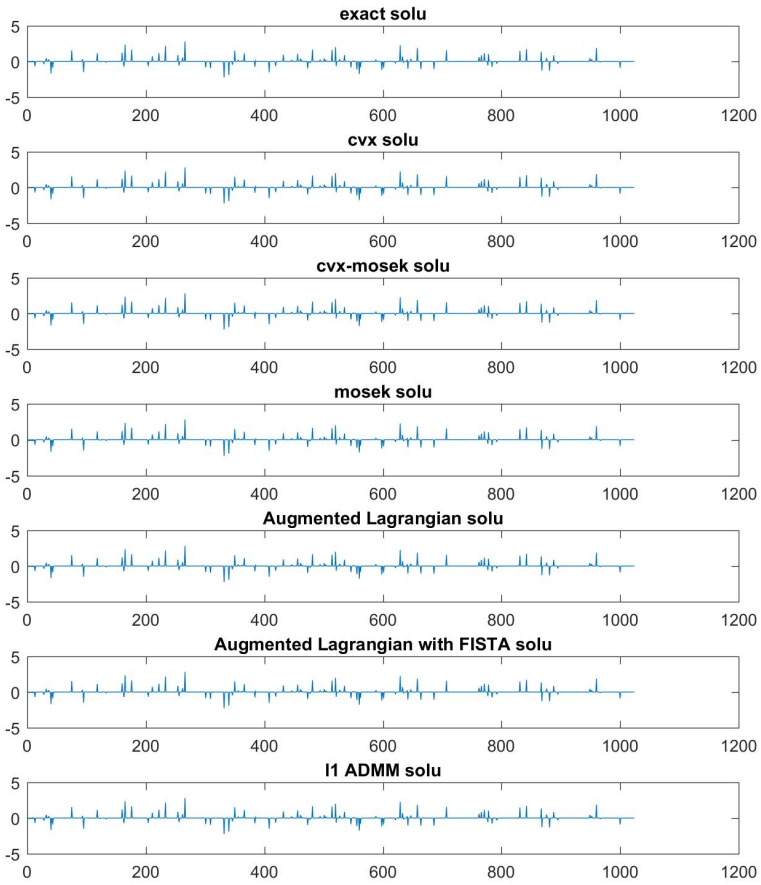

Figure 1: Qualitative Results

Method	l1 norm	residual(||Ax-b||)	err to cvx_mosek	time
CVX	8.53E+01	1.78E-08	4.40E-09	10.87
CVX with Mosek	8.53E+01	1.57E-08	0	2.49
Mosek	8.53E+01	1.59E-08	5.86E-10	2.61
Augmented Lagrangian with Proximal	8.53E+01	8.64E-03	5.56E-05	5.57
Augmented Lagrangian with FISTA	8.53E+01	5.54E-13	5.84E-10	5.61
ADMM	8.53E+01	5.05E-02	2.77E-04	0.64

Figure 2: Quantitative Results (Note the two augmented Lagrangian uses the same parameters)

Now the problem is to minimize $L_\mu(x, y)$. To do this, we use iterations. The update rule is

$$x^{k+1} := \underset{x}{\operatorname{argmin}} \; ||x||_1 + (y^k)^T(Ax - b) + \frac{1}{2\mu}||Ax - b||^2, \tag{6a}$$

$$y^{k+1} := y^k + \frac{1}{\mu}(Ax^{k+1} - b), \tag{6b}$$

To get x and y updated, we need to solve the subproblem. Here we consider two methods, i.e. proximal gradient method and accelerated proximal method(FISTA).

2.3.1 Solving the subproblem using proximal gradient methods

To solve the subproblem (6a), we first consider proximal gradient methods. Let $r(x) = \mu||x||_1$, $f(x) = \mu y^T(Ax - b) + \frac{1}{2}||Ax - b||^2$. Again, we use iterations to update our x. Each update, we have

$$x^{k+1,l} := \underset{x}{\operatorname{argmin}} \; r(x) + (\nabla f(x^{k,l}))^T(x - x^{k,l}) + \frac{1}{2\tau}||x - x^{k,l}||^2 \tag{7}$$

Here τ is another hyperparameter, controlling the proximal term, and in a way controlling the step size of each update; $x^{k,l}$ denotes that this is the k-th iteration in the augmented Lagrangian algorithm and l-th iteration in the subproblem. In each update, we minimize $r(x)$ plus first-order linearized $f(x)$ and plus proximal term. And actually (7) is equal to

$$x^{k+1,l} = soft\ thresholding(x^{k,l} - \tau\nabla f(x^{k,l}), \mu\tau) \tag{8}$$

And it's easy to calculate that $\nabla f(x) = \mu A^T y + A^T(Ax - b)$.

2.3.2 Solving the subproblem using FISTA

FISTA is similar to proximal gradient method, but is an accelerated version. The main idea is to introduce a new intermediate variable. The acceleration is shown in the complexity results that to achieve the same bound of function error, the iteration times needed is approximately the square root of ISTA or proximal gradient method. More analysis can be found in section 2.5.1 and the related literature.

The update rule of FISTA is the following: Given $y^1 = x_0$ and $t^1 = 1$, compute:

$$\begin{aligned} x_k &= soft\ thresholding(y^k - \tau\nabla f(y^k), \mu\tau) \\ t_{k+1} &= \frac{1 + \sqrt{1 + 4t_k^2}}{2} \\ y_{k+1} &= x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}) \end{aligned}$$

.

2.4 ADMM for the dual problem

In this section we derive the dual problem of (1). We also show how to use alternating direction method of multipliers(ADMM) to solve it.

2.4.1 Dual problem and its augmented Lagrangian function

Remember that the primal is

$$\min_x ||x||_1, \ s.t. \ Ax = b$$

. Therefore, the dual is

$$\max_x b^T\lambda, \ s.t. \ ||A^T\lambda||_\infty \leq 1$$

, which is equivalent to

$$\max_x b^T\lambda, \ s.t. \ A^T\lambda = s, \ ||s||_\infty \leq 1 \tag{9}$$

Therefore, as in section 2.3, we derive the augmented Lagrangian function

$$L_\mu(\lambda, s, x) := -b^T\lambda + x^T(A^T\lambda - s) + \frac{1}{2\mu}||A^T\lambda - s||^2 \tag{10}$$

2.4.2 ADMM update rule

The ADMM update rule is derived by minimizing λ, s and x one at a time. Here x is the multiplier.

$$\begin{aligned}
\lambda^{k+1} &= \operatorname*{argmin}_x L(\lambda, s^k, x^k) \\
s^{k+1} &= \operatorname*{argmin}_s L(\lambda^{k+1}, s, x^k), \ s.t. ||s||_\infty \leq 1 \\
x^{k+1} &= x^k + \frac{A^T\lambda^{k+1} - s^{k+1}}{\mu}
\end{aligned}$$

And we can derive the explicit solution of these (note that there is a typo in the slides of class, we correct it here using red):

$$\begin{aligned}
\lambda^{k+1} &= (AA^T)^{-1}(-\mu(Ax^k - b) + As^k) \\
s^{k+1} &= \mathscr{P}_{[-1,1]}(A^T\lambda^{k+1} + \mu x^k) \\
x^{k+1} &= x^k + \frac{A^T\lambda^{k+1} - s^{k+1}}{\mu}
\end{aligned}$$

Here $\mathscr{P}_{[-1,1]}(x)$ means projection of x onto $[-1, 1]$.

2.5 Results Interpretation

2.5.1 More experiments to compare proximal gradient methods and FISTA

The convergence results of proximal gradient methods(or ISTA) and FISTA are as follows: For proximal gradient method(or ISTA), we have

$$F(x_k) - F(x^*) \leq \frac{L_f ||x_0 - x^*||_2^2}{2k} \tag{11}$$

For FISTA, we have

$$F(x_k) - F(x^*) \leq \frac{L_f ||x_0 - x^*||_2^2}{(k+1)^2} \tag{12}$$

Therefore we can see that, to achieve the same error bound, the iteration times of FISTA should be much less than Proximal Gradient method. We did some experiments to validate this observation. We have two iteration parameters, one for iteration times of augmented Lagrangian method, the other for iteration times in every subproblem which is solved by proximal method or FISTA. We denote them by $iter1$ and $iter2$. We compare the efficiency and accuracy of them again:

method	iter1	iter2	error to cvx_mosek	time	res	nrm1
proximal	50	25	5.56E-05	5.56	8.64E-03	8.53E+01
FISTA	50	10	4.48E-04	2.29	6.95E-02	8.53E+01
proximal	30	40	9.69E-05	5.52	1.56E-02	8.53E+01
FISTA	30	15	7.86E-05	2.05	1.25E-02	8.53E+01
proximal	20	50	7.83E-04	5.56	1.26E-01	8.53E+01
FISTA	20	15	6.56E-04	1.65	1.04E-01	8.53E+01

Figure 3: Accuracy and Efficiency Comparison between Proximal Gradient method and FISTA

We see that though the convergence theory provides a reliable guidance for us to choose iteration times. And to achieve the same accuracy, the iteration times FISTA needs is far less than proximal method.

2.5.2 Hyperparameters

We first discuss τ and μ in augmented Lagrangian method. Despite the intuitive interpretation of τ in section 2.3.1, in the convergence proof of ISTA, we see that $\frac{1}{\tau}$ is also the Lipschitz continuous parameter L of ∇f. Since this is unknown to us, conservatively, we estimate it to be large. And therefore, τ is an extremely small value in practice.

And μ in (5) is used to control the augmented term. The smaller the μ, the larger the augmented term $||Ax - b||^2$ will weigh. Or we can also interpret μ as controller for sparsity, since the larger the μ, the larger the sparsity term $||x||_1$

will weigh. In practice, we found that μ should be relatively large. We choose μ to be 25 in our experiments.

In ADMM, we only have one hyperparameter μ, and it's also used to control the augmented term. Expectedly, it should be similar to its counterpart in augmented lagrangian method. Empirically, we found when μ is 25, the algorithm performs relatively well.

2.5.3 More on Efficiency and Accuracy

This section is an extension to the summary section 2.1 and comparison between proximal gradient and FISTA(section 2.5.1).

In the summary section, we see that in terms of residual error or error to cvxmosek, the first three methods belong to a class and the other three belong to a class. Note that cvx uses SDPT3(infeasible path-following algorithm) and both mosek and cvxmosek use interior-point method. And in terms of accuracy, they are far better than approximation algorithms like augmented Lagrangian algorithms.

As for efficiency, section 2.5.1 already shows some results. Here we take a closer look at the difference between interior point methods and approximation methods. We found that both cvxmosek and mosek solve the problem within about 2.5s. However augmented Lagrangian can be slower. One surprising discovery is that ADMM is much faster than all the other approximation methods. And its efficiency is on the same scale as cvxmosek and mosek. (Note that if we increase the iteration times, we will see that when the accuracy of ADMM is the same as mosek or cvxmosek, it has roughly the same efficiency as them, see Figure 4)

iter	residual	error to cvx_mosek	time	l1 norm
50	5.05E-02	2.77E-04	0.64	8.53E+01
100	1.11E-03	6.31E-06	1.23	8.53E+01
150	1.75E-06	1.04E-08	1.86	8.53E+01
160	6.66E-07	3.84E-09	2.02	8.53E+01
200	3.87E-09	5.88E-10	2.47	8.53E+01
Mosek	1.59E-08	5.86E-10	2.21	8.53E+01

Figure 4: more experiments of ADMM

Note when iter=160, ADMM already has the same level performance as mosek. And when iter=200, ADMM has the slightly better accuracy than mosek and is slightly slower than mosek.

3 Algorithms for Sparse Inverse Covariance Estimation

3.1 Dataset Generation

As in the reference literature [1], we let $n = 30$ and generate model1 and model2. Here we make a slight adjustment to make matrix generated from model2 to be symmetric.

3.2 Dual Problem

The primal problem is

$$max\ logdet(X) - Tr(SX) - \rho||X||_1,\ s.t. X \succeq 0 \tag{13}$$

where $||X||_1 = \sum_{ij} |X_{ij}|$. We introduce variable U, that

$$||X||_1 = \max_{||U||_\infty \leq 1} < U, X >,$$

thus transforming the original problem to

$$\max_{X \succeq 0} \min_{||U||_\infty \leq \rho} logdet(X) - < S, X > - < U, X > \tag{14}$$

where $||U||_\infty := max|U_{ij}|$.

The Lagrangian of this problem is

$$L(X, U, Y) = logdet(X) - < S + U, X > + < Y, X >, \tag{15}$$

where $Y \succeq 0$, $X \succeq 0$, $||U||_\infty \leq \rho$.

The derivative of the function if $\nabla_X L = X^{-1} - (S + U - Y)$. And therefore the dual problem is

$$min\ logdet(S + U - Y)^{-1} - n, Y \succeq 0, ||U||_\infty \leq \rho \tag{16}$$

which is equivalent to

$$min\ -logdet(S + U - Y) - n, Y \succeq 0, ||U||_\infty \leq \rho \tag{17}$$

The duality gap is

$$Duality\ gap =< S, X > +\rho||X||_1 - n \tag{18}$$

3.3 Solving Primal Using CVX

We use CVX to solve the primal problem (Equation (13)), and use duality gap to check the optimality condition. In this section, we show some exemplar numerical results in Figure 5 and Figure 6, for complete results, see section 3.6.

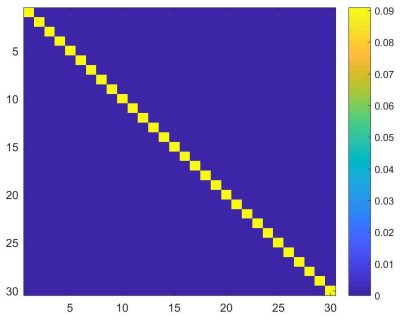

Figure 5: Model1: rho=10; duality gap=1.54e-05

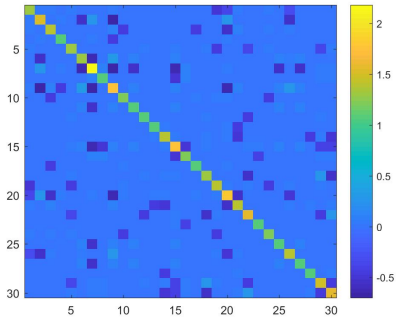

Figure 6: Model2: rho=0.1, P=30; duality gap=1.81e-04

3.4 Solving Primal using First-order Algorithm

In this section, we use the slightly modified first-order algorithm proposed in [2] to solve the primal problem. It uses the idea of Nesterov's Accelerated Gradient method([3]) and smooth minimization techniques([4]). The main advantage compared to second-order method used in cvx is it has far lower memory requirement.

Without going into too much detail of the background knowledge, we introduce the first order algorithm below: Let

$$f_\epsilon(X) := \hat{f}(X) + \max_{||U||_\infty \leq 1, U \in S^n} \{< X, U > -(\epsilon/n^2)d_2(U)\}, \tag{19}$$

where $\hat{f}(X) = -logdet(X) + < S, X >$. And let L denotes the Lipschitz constant of ∇f_ϵ.

Then the update rule is:

1. $Compute \nabla f_\epsilon(X_k) = -X^{-1} + S + U^*(X_k), where U^*(X) := max(min(X/\mu, \rho), -\rho) (which\ solves (19))$

2. $Find\ Y_k = \underset{Y \succeq 0}{\operatorname{argmin}} \{< \nabla f_\epsilon(X_k), Y - X_k > + \frac{1}{2} L ||Y - X_k||_F^2\}.$

3. $fFind\ Z_k = \underset{Z \succeq 0}{\operatorname{argmin}} \{\frac{L d_1(Z)}{\sigma 1} + \sum_{i=0}^{k} \frac{i+1}{2} (f_\epsilon(X_i) + < \nabla f_\epsilon(X_i), Z - X_i >)\}.$

4. $Update\ X_k = \frac{2}{k+3} Z_k + \frac{k+1}{k+3} Y_k\ and\ \hat{U_k} = \frac{k\hat{U_{k-1}} + 2U^*(X_k)}{k+2}.$

5. *Repeat until the duality gap is less than the target percision* :

$$-logdet(Y_k) + < S, Y_k > + \rho ||Y_k||_1 - \phi(\hat{U_k}) \leq \epsilon.$$

Here $\phi(\hat{U_k}) = \min_{X \succeq 0} -logdet(X) + < S + \hat{U_k}, X >$.

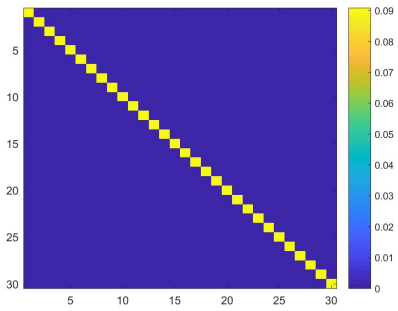

Figure 7: Model1: rho=10; duality gap=7.8e-02

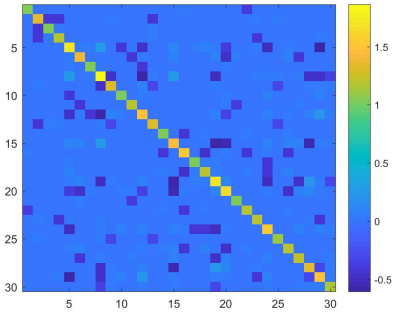

Figure 8: Model2: rho=0.1, P=30; duality gap=5e-03

Like ADMM, the Y_k, $U_k$and Z_k all have explicit solutions, which makes the algorithm practical to implement. For conciseness we don't spread all the formula here. The details can be seen in our code file $Nesterov.m$.

As we do in the last section, we also show some exemplar results here. More results can be seen in next two sections.

3.5 Comparison between CVX and First-order algorithm

We list some of our observations through numerical experiments here: 1. First-order methods suffer from much longer optimization time than cvx(which usesSDPT3)

2. First-order methods suffer from lower accuracy than cvx. My explanation is that the approximation that first-order methods use is still too inaccurate. And to achieve the theoretical convergence bound, we need a lot of iteration times. The comparison is most striking when $\rho = 10$. In Model1 case, the cvx solves the problem in about 11 seconds, while the first-order methods take much longer (about 136 seconds, and uses 77376 iterations). And even this, the first-order methods still achieve lower accuracy than cvx. This may be due to the approximation that first-order method makes.

3. According to the paper, the advantage of the first-order algorithm lies in the much lower memory requirement than cvx. However, in our case, due to the problem size, the advantage cannot be well reflected.

3.6 Complete numerical results

In this section we show the numerical results of both cvx and first-order methods for solving the problem when ρ takes different numbers. Since in model2, the condition number P has no influence on the problem, we only shows the results when $P = 30$. We show the results with the matrix duality gap and cost time

to show both the accuracy and efficiency performance of the algorithms. See Figures 9 to 20.

References

[1] Tony Cai, Weidong Liu, and Xi Luo. A constrained l_1 minimization approach to sparse precision matrix estimation. *Journal of the American Statistical Association*, 106(494):594–607, 2011.

[2] Alexandre d'Aspremont, Onureena Banerjee, and Laurent El Ghaoui. First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30(1):56–66, 2008.

[3] Y. NESTEROV. A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. *Doklady AN USSR*, 269:543–547, 1983.

[4] Yu. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, May 2005.

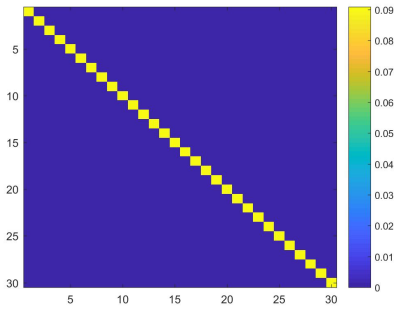

Figure 9: CVX-Model1: rho=10; duality gap=1.54e-05; time=11.74sec

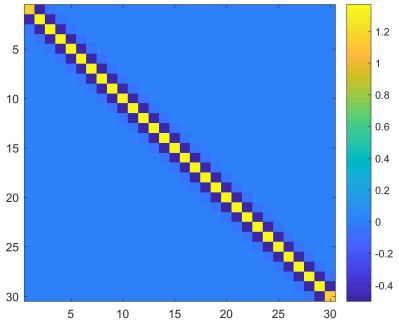

Figure 10: CVX-Model1: rho=0.1; duality gap=2.08e-04; time=10.97

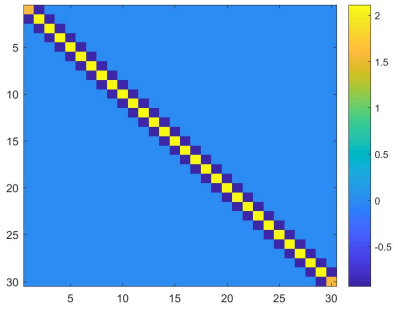

Figure 11: CVX-Model1: rho=0.001; duality gap=9.22e-05; time=8.68sec

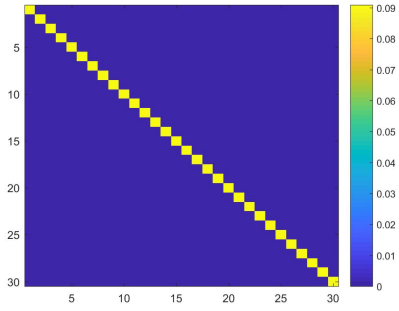

Figure 12: CVX-Model2: rho=10; duality gap=1.03e-05; time=13.18

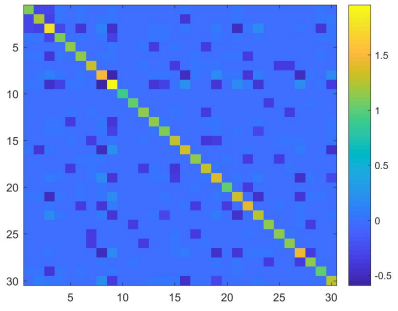

Figure 13: CVX-Model2: rho=0.1; duality gap=1.87e-04; time=11.51sec

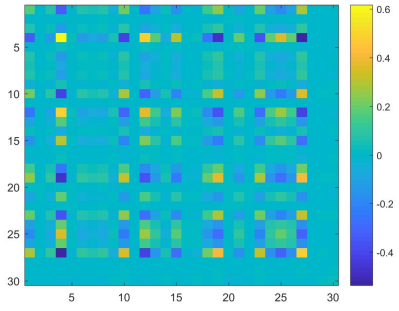

Figure 14: CVX-Model2: rho=0.001; duality gap=31.00; unbounded

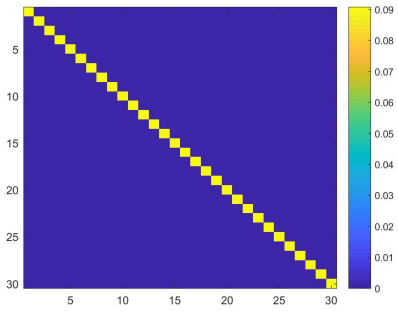

Figure 15: FirstOrder-Model1: rho=10; duality gap=7.87e-02; time=92.86sec iter=50236

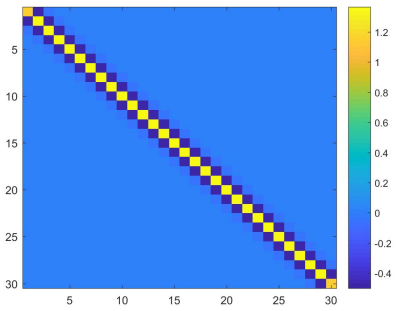

Figure 16: FirstOrder-Model1: rho=0.1; duality gap=5e-3; time=114.6sec iter=65882

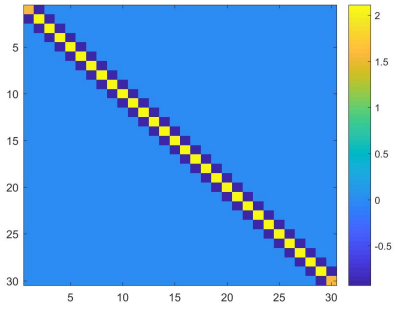

Figure 17: FirstOrder-Model1: rho=0.001; duality gap=1e-03; time=47.35sec iter=25758

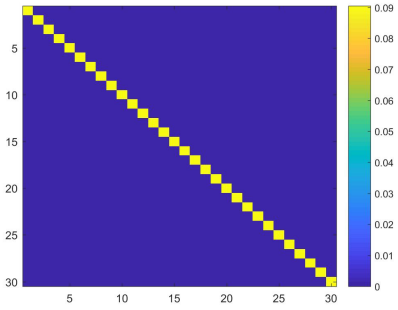

Figure 18: FirstOrder-Model2: rho=10; duality gap=5.35e-2; time=31.74sec iter=13470

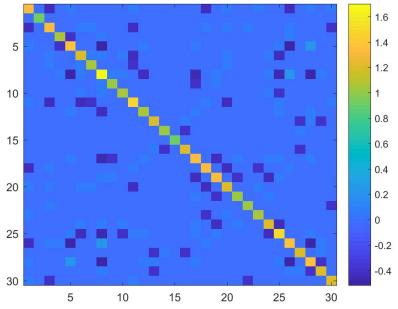

Figure 19: FirstOrder-Model2: rho=0.1; duality gap=5e-03; time=113.85sec iter=63562

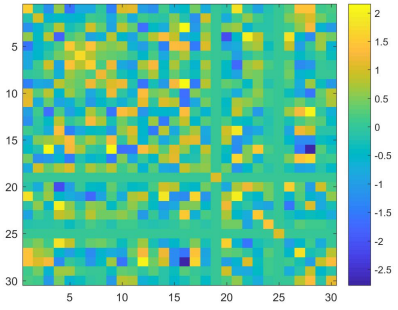

Figure 20: The inverse of Model2: rho=0.001; This case has no feasible point (doesn't converge)