ADMM Techniques for Linear & Semidefinite Programming

Nico Chaves, Mike Schoenhals, Junjie (Jason) Zhu

March 18, 2017

1 Introduction

The alternating direction method of multipliers (ADMM) is a 1st order optimization method. In this project, we experimented with variants of ADMM applied to the following optimization problems.
Linear Programming Primal:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{LP}$$

Linear Programming Dual:

$$\begin{aligned} \max_{\mathbf{y},\mathbf{s}} \quad & \mathbf{b}^T\mathbf{y} \\ \text{subject to} \quad & A^T\mathbf{y} + \mathbf{s} = \mathbf{c}, \\ & \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{LD}$$

SDP Primal:

$$\begin{aligned} \min_{X} \quad & C \bullet X \\ \text{s.t.} \quad & \mathcal{A}X = \mathbf{b} \\ & X \succeq \mathbf{0} \end{aligned} \tag{SDP}$$

SDP Dual:

$$\begin{aligned} \max_{\mathbf{y},S} \quad & \mathbf{b}^T\mathbf{y} \\ \text{s.t.} \quad & \mathcal{A}^T\mathbf{y} + S = C \\ & S \succeq \mathbf{0} \end{aligned} \tag{SDD}$$

For (LP) and (LD), we also experimented with block splitting, interior point methods, and preconditioning. Our solvers and experiments are available online: https://github.com/nmchaves/admm-for-lp.

2 Basic ADMM for Linear Programming

In this section, we derive the ADMM update procedures for solving (LP) and (LD).

Primal Update

We re-formulate the primal LP problem as:

$$\begin{aligned} \text{minimize}_{\mathbf{x}_1,\mathbf{x}_2} \quad & \mathbf{c}^T\mathbf{x}_1 \\ \text{subject to} \quad & A\mathbf{x}_1 = \mathbf{b}, \\ & \mathbf{x}_1 - \mathbf{x}_2 = \mathbf{0}, \\ & \mathbf{x}_2 \geq \mathbf{0} \end{aligned} \tag{LPR}$$

and consider the augmented Lagrangian function:

$$L^P(\mathbf{x}_1,\mathbf{x}_2,\mathbf{y},\mathbf{s}) = \mathbf{c}^T\mathbf{x}_1 - \mathbf{y}^T\left(A\mathbf{x}_1-\mathbf{b}\right) - \mathbf{s}^T\left(\mathbf{x}_1-\mathbf{x}_2\right) + \frac{\beta}{2}\left(\|A\mathbf{x}_1-\mathbf{b}\|^2 + \|\mathbf{x}_1-\mathbf{x}_2\|^2\right)$$

The decision variable updates are given by (see Appendix for derivation):

$$\mathbf{x}_1^{k+1} = \left(A^TA+I\right)^{-1}\left(\frac{1}{\beta}A^T\mathbf{y}^k + \frac{1}{\beta}\mathbf{s}^k - \frac{1}{\beta}\mathbf{c} + A^T\mathbf{b}^k + \mathbf{x}_2^k\right) \tag{1}$$

$$x_{2,j}^{k+1} = \max\left\{x_{1,j}^{k+1} - \frac{1}{\beta}s_j^k, 0\right\} \text{ for } j = 1,..,n. \tag{2}$$

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \beta(A\mathbf{x}_1^{k+1} - \mathbf{b}) \tag{3}$$

$$\mathbf{s}^{k+1} = \mathbf{s}^k - \beta(\mathbf{x}_1^{k+1} - \mathbf{x}_2^{k+1}) \tag{4}$$

We terminate when the solution is primal feasible, i.e., $A\mathbf{x}_1^{k+1} - \mathbf{b} \approx \mathbf{0}$ and $\mathbf{x}_1^{k+1} - \mathbf{x}_2^{k+1} \approx \mathbf{0}$.

Dual Update

We reformulate (LD) as a minimization problem:

$$\begin{aligned} \text{minimize}_{\mathbf{y},\mathbf{s}} \quad & -\mathbf{b}^T\mathbf{y} \\ \text{subject to} \quad & A^T\mathbf{y} + \mathbf{s} = \mathbf{c}, \\ & \mathbf{s} \geq \mathbf{0} \end{aligned} \tag{LDR}$$

The dual's augmented Lagrangian function is then:

$$L^d(\mathbf{y},\mathbf{s},\mathbf{x}) = -\mathbf{b}^T\mathbf{y} - \mathbf{x}^T\left(A^T\mathbf{y}+\mathbf{s}-\mathbf{c}\right) + \frac{\beta}{2}\left\|A^T\mathbf{y}+\mathbf{s}-\mathbf{c}\right\|^2.$$

The decision variable updates are given by:

$$\mathbf{y}^{k+1} = \left(AA^T\right)^{-1}\left(\frac{1}{\beta}\left(A\mathbf{x}^k + \mathbf{b}\right) - A\mathbf{s}^k + A\mathbf{c}\right), \tag{5}$$

$$s_j^{k+1} = \max\left\{\frac{1}{\beta}x_j^k - (A^T\mathbf{y}^{k+1})_j + c_j, 0\right\} \text{ for } j = 1,..,n. \tag{6}$$

We update $\mathbf{x}$ using

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \beta\left(A^T\mathbf{y}^{k+1} + \mathbf{s}^{k+1} - \mathbf{c}\right). \tag{7}$$

We terminate and declare optimality when $A\mathbf{x}^{k+1} + \mathbf{b} \approx \mathbf{0}$. Note: since we converted the problem to a minimization, the final $\mathbf{x}$ value will be non-positive. As a result, we must negate it to obtain the solution to the original problem.

3 Interior-Point ADMM for LP

Interior Point Primal LP Updates

Here we modify the formulation in (LPR) using the log barrier function:

$$\begin{aligned} \min_{\mathbf{x}_1,\mathbf{x}_2} \quad & \mathbf{c}^T\mathbf{x}_1 - \mu\sum_j \ln(x_{2,j}) \\ \text{subject to} \quad & A\mathbf{x}_1 = \mathbf{b}, \\ & \mathbf{x}_1 - \mathbf{x}_2 = \mathbf{0} \end{aligned} \tag{LPRB}$$

The constraint $\mathbf{x}_2 > \mathbf{0}$ is now implicit. The augmented Lagrangian function is given by

$$L^p_\mu(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}, \mathbf{s}) = \mathbf{c}^T\mathbf{x}_1 - \mu \sum_j \ln(x_{2,j}) - \mathbf{y}^T(A\mathbf{x}_1 - \mathbf{b}) - \mathbf{s}^T(\mathbf{x}_1 - \mathbf{x}_2) + \frac{\beta}{2}\left(\|A\mathbf{x}_1 - \mathbf{b}\|^2 + \|\mathbf{x}_1 - \mathbf{x}_2\|^2\right).$$

The $\mathbf{x}_1$ update is the same as (16). The 1st order gradient condition for $\mathbf{x}_2$ is

$$s_j - \frac{\mu}{x_{2,j}} + \beta(x_{2,j} - x_{1,j}) = 0 \tag{8}$$

which yields the update

$$x_{2,j} = \frac{1}{2\beta}\left(\beta x_{1,j} - s_j + \sqrt{\beta^2 x_{1,j}^2 - 2\beta s_j x_{1,j} + s_j^2 + 4\beta\mu}\right), \text{ for } j = 1, .., n \tag{9}$$

The update equations for $\mathbf{y}$ and $\mathbf{s}$ remain the same as in (18) and (19). At the end of each iteration, we update $\mu^{k+1} = \gamma\mu^k$ for some constant $\gamma < 1$, and the termination condition remains the same as the non-interior-point method.

Interior Point Dual LP Updates

We modify (LDR) using the log barrier function:

$$\begin{aligned} \min_{\mathbf{y},\mathbf{s}} \quad & -\mathbf{b}^T\mathbf{y} - \mu\sum_j \ln(s_j) \\ \text{subject to} \quad & A^T\mathbf{y} + \mathbf{s} = \mathbf{c} \end{aligned} \tag{LDRB}$$

The constraint $\mathbf{s} > \mathbf{0}$ is now implicit. The augmented Lagrangian function is given by

$$L^d_\mu(\mathbf{y}, \mathbf{s}, \mathbf{x}) = -\mathbf{b}^T\mathbf{y} - \mu\sum_j \ln(s_j) - \mathbf{x}^T\left(A^T\mathbf{y} + \mathbf{s} - \mathbf{c}\right) + \frac{\beta}{2}\left\|A^T\mathbf{y} + \mathbf{s} - \mathbf{c}\right\|^2.$$

The $\mathbf{y}$ update remains the same as (24). Using the 1st order gradient condition for $\mathbf{s}$

$$-\frac{\mu}{s_j} + x_j + \beta\left(s_j - (\mathbf{c} - (A^T\mathbf{y}))_j\right) = 0 \tag{10}$$

we obtain the $\mathbf{s}$ update

$$s_j = \frac{1}{2\beta}\left(\beta(\mathbf{c} - (A^T\mathbf{y}))_j + x_j + \sqrt{\beta^2(\mathbf{c} - (A^T\mathbf{y}))_j^2 - 2\beta(\mathbf{c} - (A^T\mathbf{y}))_j x_j + x_j^2 + 4\beta\mu}\right), \text{ for } j = 1, .., n \tag{11}$$

The update equation for $\mathbf{x}$ remains the same as in (7). As in the primal interior-point algorithm, we update $\mu^{k+1} = \gamma\mu^k$ for some constant $\gamma < 1$ at the end of each iteration, and the termination condition remains the same as the non-interior-point method.

4 Multi-Block ADMM for LP

As shown in the previous section, certain ADMM update steps require computation of a potentially large matrix inverse. Tthis computation may become the bottleneck for sufficiently large problems. By splitting the decision variables into blocks and updating the blocks sequentially, one can reduce the runtime by computing multiple small matrix inverses instead of a single large matrix inverse.

However, when the decision variables are separable, it may not be necessary to apply block updates on them. At the end of each iteration, these separable variables would take identical values no matter how they are grouped. Thus, we only need to consider the cases where we split $\mathbf{x_1}$ in the primal problems (i.e., (LPR) and (LPRB)), or to split $\mathbf{y}$ in the dual problems (i.e., (LDR) and (LDRB)).

Primal ADMM with Block Splitting

Consider splitting the problem into B blocks of equal size as follows:

$$\mathbf{x}_1 = \begin{bmatrix} \mathbf{x}_{1,1} \\ \mathbf{x}_{1,2} \\ \vdots \\ \mathbf{x}_{1,B} \end{bmatrix}, \quad A = \begin{bmatrix} A_1 & A_2 & \dots & A_B \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_B \end{bmatrix}$$

Let U denote the set of blocks which have already been updated at the current iteration. To update block i of $\mathbf{x}_1$, we compute:

$$\mathbf{x}_{1,i}^{k+1} = \left(A_i^T A_i + I\right)^{-1} \left(\frac{1}{\beta} A_i^T \mathbf{y} + \frac{1}{\beta}\mathbf{s}_i - \frac{1}{\beta}\mathbf{c}_i + A_i^T \mathbf{b} + \mathbf{x}_{2,i}^k - \sum_{j \neq i, j \in U} A_i^T A_j \mathbf{x}_{1,j}^{k+1} - \sum_{j \neq i, j \notin U} A_i^T A_j \mathbf{x}_{1,j}^k \right)$$

where A_i refers to the i^{th} block of columns of A. See the Appendix for a derivation.

Dual ADMM with Block Splitting

TODO: need to show how the blocks get split!!

Let U denote the set of blocks which have already been updated at the current iteration. To update block i of $\mathbf{y}$:

$$\mathbf{y}_i^{k+1} = \left(A_i A_i^T\right)^{-1} \left(\frac{1}{\beta}\left(A_i \mathbf{x} + \mathbf{b}_i\right) - A_i\left(\mathbf{s} - \mathbf{c}\right) - \sum_{j \neq i, j \in U} A_i A_j^T \mathbf{y}_j^{k+1} - \sum_{j \neq i, j \notin U} A_i A_j^T \mathbf{y}_j^k \right)$$

See the Appendix for a derivation.

5 Preconditioning

In this report, we demonstrate that preconditioning can help ADMM solve linear programs faster. After preconditioning the system, we solve the problem using the modified constraints:

$$A' x = \mathbf{b}' \tag{12}$$

where A' and $\mathbf{b}'$ are the results of preconditioning A and $\mathbf{b}$, respectively.

Standard Preconditioning

To apply "standard" preconditioning to a linear programming problem, one first computes

$$A' = (AA^T)^{-1/2} A, \qquad \mathbf{b}' = (AA^T)^{-1/2}\mathbf{b} \tag{13}$$

and then substitutes A' and $\mathbf{b}'$ for A and $\mathbf{b}$ in the original problems (LP) and (LD). Clearly, this reformulation does not change the feasible regions for (LP) or (LD).

Cholesky Preconditioning

We can also precondition our system using a Cholesky factorization:

$$AA^T = LL^T, \qquad A' = L^{-1}A, \qquad \mathbf{b}' = L^{-1}\mathbf{b}$$

where L is a lower triangular matrix. To perform the preconditioning more quickly, we can use an incomplete Cholesky factorization:

$$A \approx \hat{L}\hat{L}^T, \qquad A' = \hat{L}^{-1}A, \qquad \mathbf{b}' = \hat{L}^{-1}\mathbf{b}$$

6 ADMM for SDP Cones

ADMM for SDP Primal

Similar to the LP case, we divide X into X_1 and X_2 and solve the reformulated problem:

$$\begin{aligned}\min_{X_1,X_2} \quad & C \bullet X_1 \\ \text{s.t.} \quad & \mathcal{A}X_1 = b \\ & X_1 = X_2 \\ & X_2 \succeq 0\end{aligned}$$

Augmented Lagrangian:

$$L = C \bullet X_1 - y^T\left(\mathcal{A}X_1 - b\right) - Z \bullet \left(X_1 - X_2\right) + \frac{\beta}{2}\left\|\mathcal{A}X_1 - b\right\|^2 + \frac{\beta}{2}\left\|X_1 - X_2\right\|_f^2$$

The decision variable updates are then (see Appendix for the derivation):

$$X_1 = \left(\mathcal{A}^T\mathcal{A} + I\right)^{-1}\left(X_2 + \frac{1}{\beta}\left(\mathcal{A}^T y + Z - C\right) + \mathcal{A}^T b\right)$$

$$X_2 = \max\{0, -\frac{Z}{\beta} - X_1\}$$

Here, the operation $\max\{0, M\}$ means to set any negative eigenvalues of M to 0. In particular, consider the eigendecomposition of M: $M = VDV^{-1}$. We compute $\hat{D} = max\{0, D\}$ (this max is computed element-wise), then set $X_2 = V\hat{D}V^{-1}$. The dual variable updates follow the LP case very closely, except with s replaced by the matrix S.

ADMM for SDP Dual

The augmented Lagrangian for (SDD) is

$$L = \mathbf{b}^T\mathbf{y} - X \bullet \left(\mathcal{A}^T\mathbf{y} + S - C\right) + \frac{\beta}{2}\left\|\mathcal{A}^T\mathbf{y} + S - C\right\|_f^2$$

The decision variable updates are (see Appendix for the derivation):

$$y = \left(\mathcal{A}\mathcal{A}^T\right)^{-1}\left(\frac{1}{\beta}\left(\mathcal{A}X - b\right) + \mathcal{A}\left(C - S\right)\right)$$

$$S = \max\left\{0, \frac{1}{\beta}X + C - \mathcal{A}^T y\right\}$$

Here, $\max\{0, M\}$ is defined in the SDP primal section above.
The X update is similar to the LP case: $X^{k+1} = X^k - \beta\left(\mathcal{A}^T\mathbf{y}^{k+1} + S^{k+1} - C\right)$

7 Experiments and Results

We evaluated the performance of the different ADMM approaches by randomly simulating problems and recording the number of iterations needed to converge. Each set of experiments is described in more detail below.

Preconditioning with no Block-Splitting

First, we studied how preconditioning affects the convergence rate of the basic primal and dual ADMM algorithms in (LPR) and (LDR). For simplicity, we did not apply block splitting. We evaluated the two algorithms over 10 random problems of size 50×300.

We evaluated how sensitive these ADMM LP algorithms are to the β parameter in the augmented Lagrangian. For each algorithm, we recorded the number of iterations required to converge as a function of β. The results are shown in Figure 1. We observe that all of the ADMM approaches are relatively insensitive to β. For the rest of our experiments, we fixed $\beta = 0.9$. Choosing a different value of β should not have much effect on the rest of our results.

Additionally, we notice that the basic primal ADMM implementation converges in fewer iterations than the basic dual ADMM, but when preconditioning is applied (on the same problems), the dual converges in fewer iterations than the primal.

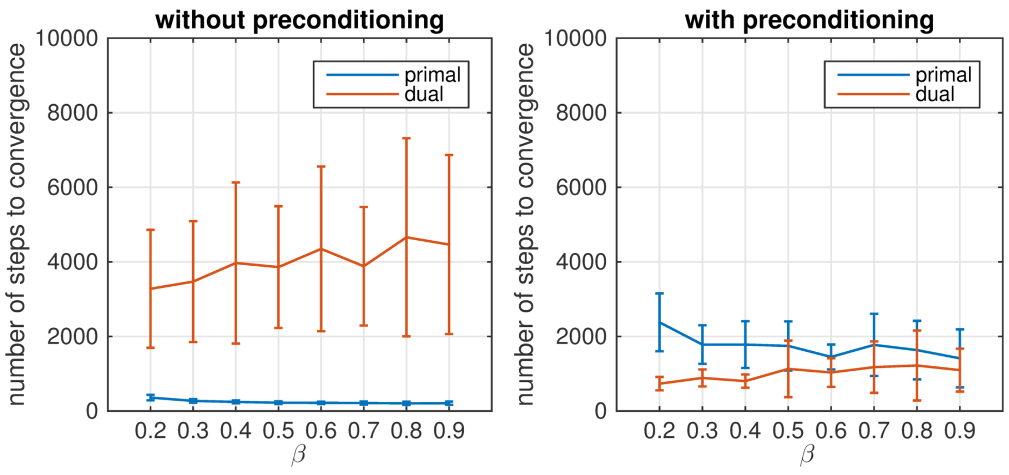

Figure 1: Results on 10 random LP problem of size 50×300.

Baseline Block-splitting (No Random Permutation, No Preconditioning)

To establish a baseline, we evaluated the ADMM solvers using sequential updating (i.e. we did not randomly permute the block update order) and without using preconditioning. The results of this experiment are shown in Figure 2. In the figure, B represents the number of blocks ($B = 1$ indicates that no splitting was performed). Note that when $B > 2$, the problem does not necessarily converge. As we will see in the following experiments, we can improve performance by introducing random permutations or preconditioning.

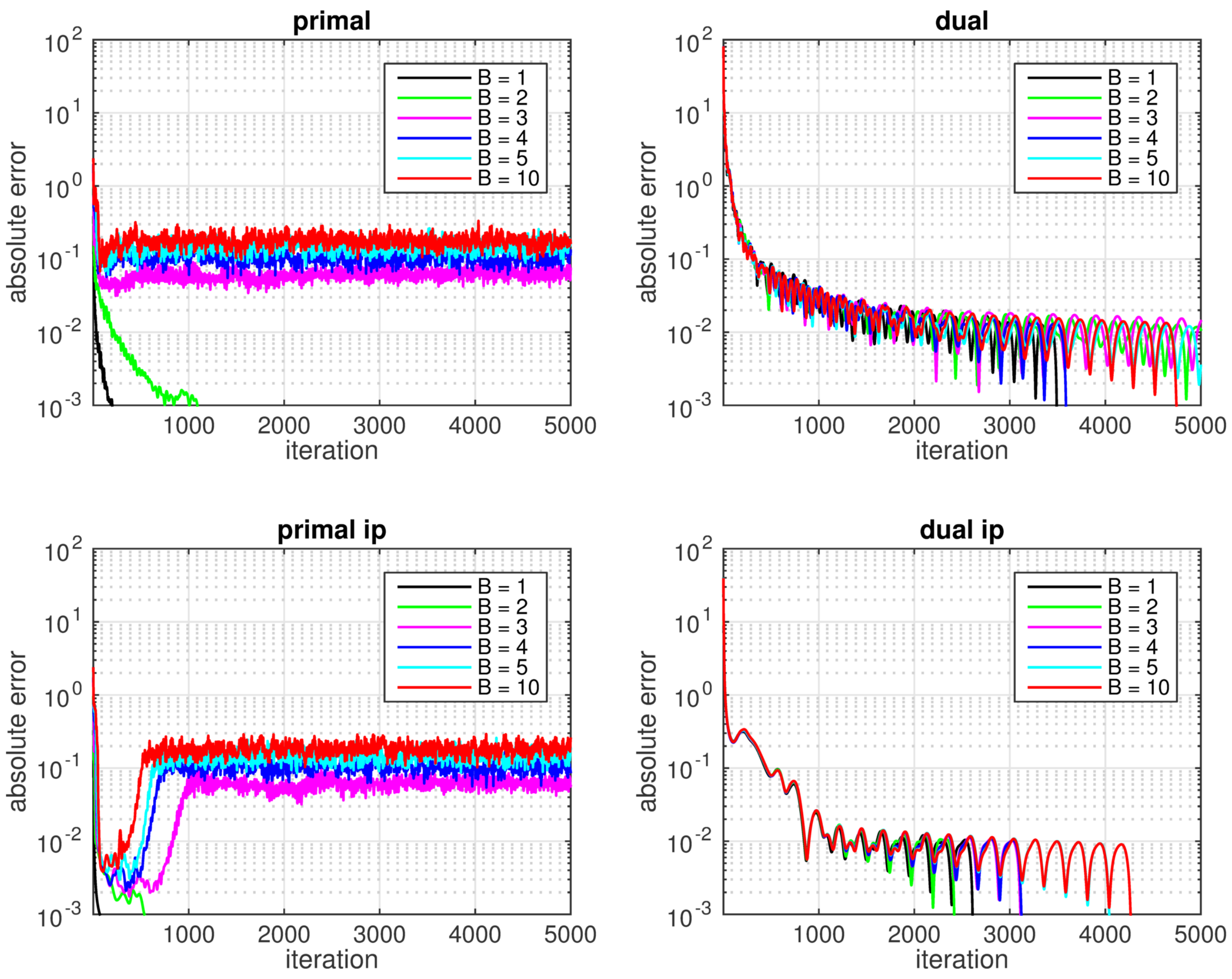

Figure 2: Results on a random LP problem of size 50×300. No preconditioning and no random permutation.

Block-splitting with Random Permutation (No Preconditioning)

In this set of experiments, we randomly permuted the update order of the block variables at each iteration. The results are shown in Figure 3. When using random permutation, the primal now converges for all block sizes. Note that splitting the primal into more blocks requires more iterations. However, performing block splitting may still result in lower overall runtime because of the faster inverse computations.

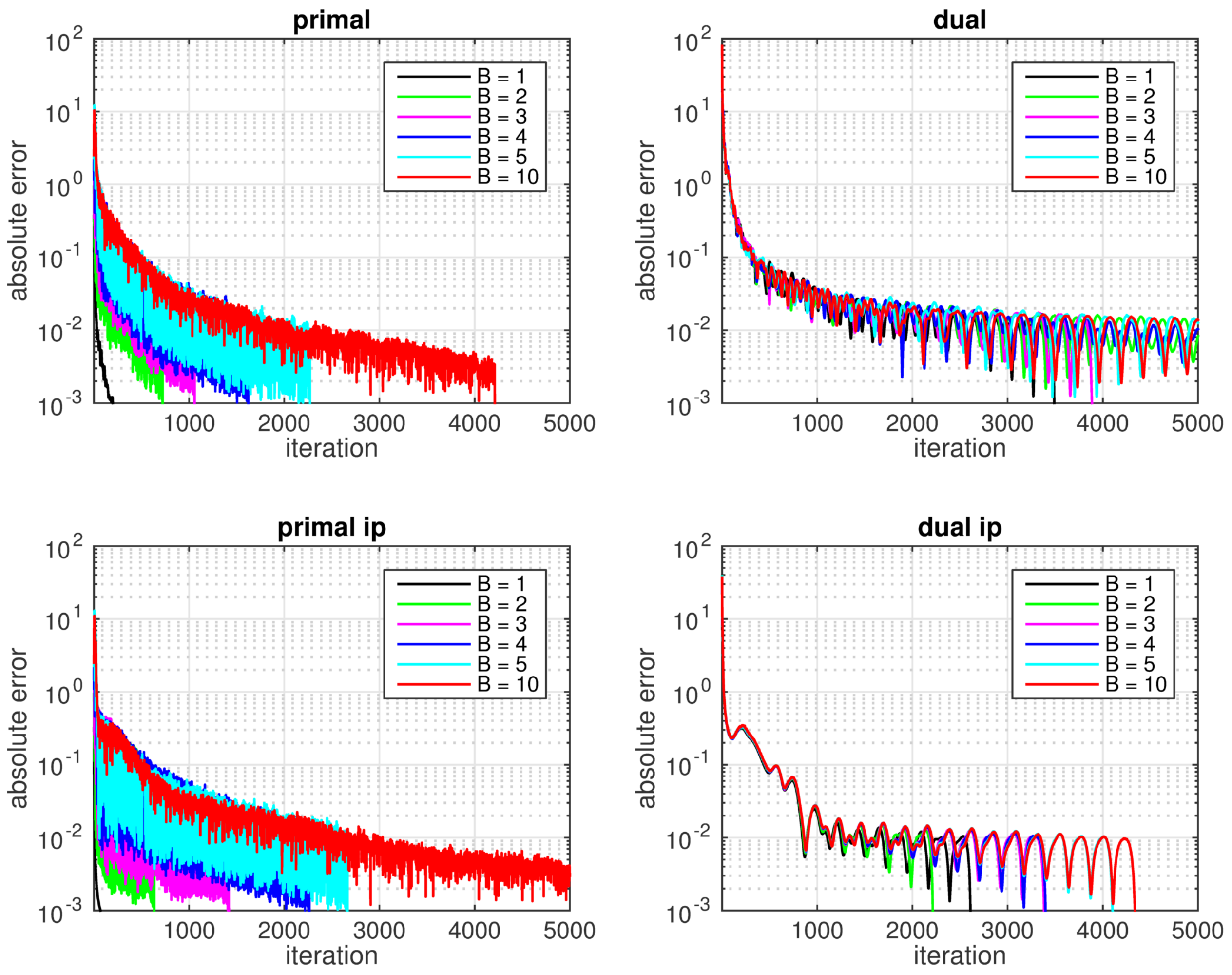

Figure 3: Results on a random LP problem of size 50×300. Only random permutation and no preconditioning.

Block-splitting with Preconditioning (No Random Permutation)

In this set of experiments, we applied preconditioning to each problem and used sequential block updates (no random permutation). The results are shown in Figure 4. Note that the number of blocks has no effect on the number of iterations required for either dual algorithm. This verifies our claim at the end of Section 3 that preconditioning applied to the dual causes $\mathbf{y}$ to be separable. Interestingly, the number of blocks has very little effect on the number of iterations for the primal.

These results suggest that one should prefer to use the maximum number of blocks possible (n for the primal and m for the dual) when using preconditioning. In this case, the matrices that need to be inverted in the decision variable updates become scalars, so the updates can be computed rapidly.

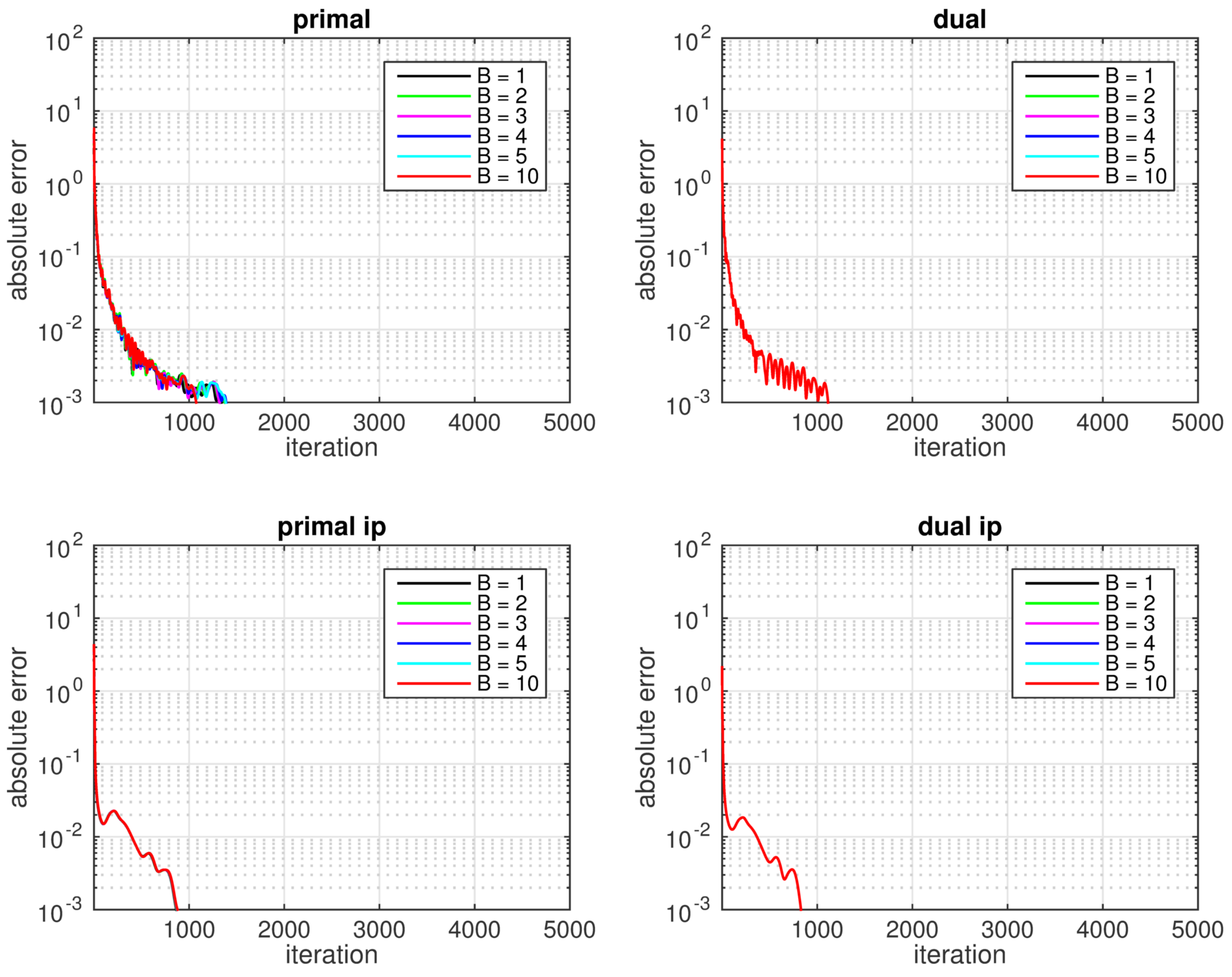

Figure 4: Results on a random LP problem of size 50×300. Only preconditioning and no random permutation.

8 Conclusions and Future Work

We evaluated the performance of several different versions of ADMM for linear programming problems. Our experiments show that one can split the problem into blocks without significantly increasing the number of iterations required to find an optimal solution. However, one must be sure to apply preconditioning to the problem or use a randomly permuted update order. Since block splitting allows one to avoid a large matrix inversion, our results suggest that these techniques may reduce the overall runtime of ADMM algorithms. We also found that block splitting seems especially well suited to problems with an inherent block structure, which commonly arise in practice.

In our future work, we intend to run similar experiments on more problems to verify that the results are robust. We also intend to test each approach on problems with various sizes. It will be interesting to see how each approach scales as problems grow large. We plan to compare the runtime (not just the number of iterations) of each algorithm, accounting for the time to take large matrix inversions. This will enable us to determine whether avoiding the large matrix inverse actually leads to a speedup. Finally, we intend to experiment with inverse-free preconditioning methods, such as Cholesky preconditioning, to avoid the matrix inversion involved in our current preconditioning approach. Indeed, we have shown cases where LP problems cannot be solved correctly without preconditioning or random-permutation on the block updates, and the results we present here on preconditioning make inverse-free preconditioning methods promising for multi-block ADMM.

9 Acknowledgements

We'd like to thank Professor Yinyu Ye for his mentorship and guidance throughout the project.

10 Appendix

LP Primal Update Derivation

To solve (**??**), we update $\mathbf{x}_1$ and $\mathbf{x}_2$ sequentially using:

$$\mathbf{x}_1^{k+1} = \arg\min_{\mathbf{x}_1^k} L^P(\mathbf{x}_1^k, \mathbf{x}_2^k, \mathbf{y}^k, \mathbf{s}^k), \tag{14}$$

$$\mathbf{x}_2^{k+1} = \arg\min_{\mathbf{x}_2^k \geq 0} L^P(\mathbf{x}_1^{k+1}, \mathbf{x}_2^k, \mathbf{y}^k, \mathbf{s}^k). \tag{15}$$

Using the 1st order gradient condition

$$\nabla_{\mathbf{x}_1} L^P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}, \mathbf{s}) = \mathbf{c} - A^T\mathbf{y} - \mathbf{s} + \beta\left(A^T\left(A\mathbf{x}_1 - \mathbf{b}\right) + (\mathbf{x}_1 - \mathbf{x}_2)\right) = \mathbf{0}$$

we obtain the update step for $\mathbf{x}_1$

$$\mathbf{x}_1^{k+1} = \left(A^TA + I\right)^{-1}\left(\frac{1}{\beta}A^T\mathbf{y}^k + \frac{1}{\beta}\mathbf{s}^k - \frac{1}{\beta}\mathbf{c} + A^T\mathbf{b}^k + \mathbf{x}_2^k\right) \tag{16}$$

Similarly, we solve

$$\nabla_{\mathbf{x}_2} L^P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}, \mathbf{s}) = \mathbf{s} + \beta\left(\mathbf{x}_2 - \mathbf{x}_1\right) = \mathbf{0}$$

to obtain the update for $\mathbf{x}_2$. Since the elements of $\mathbf{x}_2$ are separable, we can update each one independently as follows:

$$x_{2,j}^{k+1} = \max\left\{x_{1,j}^{k+1} - \frac{1}{\beta}s_j^k, 0\right\} \text{ for } j = 1, .., n. \tag{17}$$

Next, we update the multipliers $\mathbf{y}$ and $\mathbf{s}$ using steepest ascent:

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \beta(A\mathbf{x}_1^{k+1} - \mathbf{b}) \tag{18}$$

$$\mathbf{s}^{k+1} = \mathbf{s}^k - \beta(\mathbf{x}_1^{k+1} - \mathbf{x}_2^{k+1}) \tag{19}$$

LP Dual Update Derivation

Similar to the primal updates, we update $\mathbf{y}$ and $\mathbf{s}$ according to

$$\mathbf{y}^{k+1} = \arg\min_{\mathbf{y}} L^d(\mathbf{y}^k, \mathbf{s}^k, \mathbf{x}^k), \tag{20}$$

$$\mathbf{s}^{k+1} = \arg\min_{\mathbf{s}\geq 0} L^d(\mathbf{y}^{k+1}, \mathbf{s}^k, \mathbf{x}^k). \tag{21}$$

Using the 1st order gradient conditions

$$\nabla_{\mathbf{y}} L^d(\mathbf{y}, \mathbf{s}, \mathbf{x}) = -\mathbf{b} - A\mathbf{x} + \beta A\left(A^T\mathbf{y} + \mathbf{s} - \mathbf{c}\right) = \mathbf{0}, \tag{22}$$

$$\nabla_{\mathbf{s}} L^d(\mathbf{y}, \mathbf{s}, \mathbf{x}) = -\mathbf{x} + \beta\left(A^T\mathbf{y} + \mathbf{s} - \mathbf{c}\right) = \mathbf{0}, \tag{23}$$

we have the updates for $\mathbf{y}$ and $\mathbf{s}$:

$$\mathbf{y}^{k+1} = \left(AA^T\right)^{-1}\left(\frac{1}{\beta}\left(A\mathbf{x}^k + \mathbf{b}\right) - A\mathbf{s}^k + A\mathbf{c}\right), \tag{24}$$

$$s_j^{k+1} = \max\left\{\frac{1}{\beta}x_j^k - (A^T\mathbf{y}^{k+1})_j + c_j, 0\right\} \text{ for } j = 1, .., n. \tag{25}$$

Primal ADMM with Block Splitting (B=2 Blocks)

To help make our derivations concrete, we first consider splitting the problem into $B = 2$ blocks of equal size as follows:

$$\mathbf{x}_1 = \begin{bmatrix}\mathbf{x}_{1,1}\\ \mathbf{x}_{1,2}\end{bmatrix}, \quad A = \begin{bmatrix}A_1 & A_2\end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix}\mathbf{c}_1\\ \mathbf{c}_2\end{bmatrix}$$

Then the Lagrangian can be expressed as:

$$\begin{aligned} L^P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}) \quad = \quad & \mathbf{c}_1^T\mathbf{x}_{1,1} + \mathbf{c}_2^T\mathbf{x}_{1,2} - \mathbf{y}^T\left(A_1\mathbf{x}_{1,1} + A_2\mathbf{x}_{1,2} - \mathbf{b}\right) - \mathbf{s}_1^T\left(\mathbf{x}_{1,1} - \mathbf{x}_{2,1}\right) - \mathbf{s}_2^T\left(\mathbf{x}_{1,2} - \mathbf{x}_{2,2}\right) \\ & + \frac{\beta}{2}\left(\|A_1\mathbf{x}_{1,1} + A_2\mathbf{x}_{1,2} - \mathbf{b}\|^2 + \|\mathbf{x}_{1,1} - \mathbf{x}_{2,1}\|^2 + \|\mathbf{x}_{1,2} - \mathbf{x}_{2,2}\|^2\right) \end{aligned}$$

Taking the gradient with respect to the 1st block of $\mathbf{x}_1$:

$$\nabla_{\mathbf{x}_{1,1}} L^P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}) = \mathbf{c}_1 - A_1^T\mathbf{y} - \mathbf{s}_1 + \beta\left(A_1^T\left(A_1\mathbf{x}_{1,1} + A_2\mathbf{x}_{1,2} - \mathbf{b}\right) + \left(\mathbf{x}_{1,1} - \mathbf{x}_{2,1}\right)\right)$$

Setting the gradient to 0, we obtain the update for $\mathbf{x}_{1,1}$:

$$\mathbf{x}_{1,1}^{k+1} = \left(A_1^TA_1 + I\right)^{-1}\left(\frac{1}{\beta}A_1^T\mathbf{y} + \frac{1}{\beta}\mathbf{s}_1 - \frac{1}{\beta}\mathbf{c}_1 + A_1^T\mathbf{b} + \mathbf{x}_{2,1}^k - A_1^TA_2\mathbf{x}_{1,2}^k\right)$$

By symmetry, the update step for the 2nd block of $\mathbf{x}_1$ is:

$$\mathbf{x}_{1,2}^{k+1} = \left(A_2^TA_2 + I\right)^{-1}\left(\frac{1}{\beta}A_2^T\mathbf{y} + \frac{1}{\beta}\mathbf{s}_2 - \frac{1}{\beta}\mathbf{c}_2 + A_2^T\mathbf{b} + \mathbf{x}_{2,2}^k - A_2^TA_1\mathbf{x}_{1,1}^{k+1}\right)$$

Note that here we use $\mathbf{x}_{1,1}^{k+1}$ to update $\mathbf{x}_{1,2}$. Our results show that when $B > 2$ blocks, one may need to randomly permute the update order. As a result, on some iterations, we may first update $\mathbf{x}_{1,2}$, then use the new value to update $\mathbf{x}_{1,1}$.

Dual ADMM with Block Splitting (B=2 Blocks)

Now, we derive the corresponding update for the dual ADMM with block splitting. Again, we first consider the $B = 2$ case and split the problem into 2 blocks of equal size:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \quad A^T = \begin{bmatrix} A_1^T & A_2^T \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

Then the Lagrangian can be expressed as:

$$L^d(\mathbf{y}, \mathbf{s}, \mathbf{x}) = -\mathbf{b}_1^T \mathbf{y}_1 - \mathbf{b}_2^T \mathbf{y}_2 - \mathbf{x}^T \left(A_1^T \mathbf{y}_1 + A_2^T \mathbf{y}_2 + \mathbf{s} - \mathbf{c} \right) + \frac{\beta}{2} \left\| A_1^T \mathbf{y}_1 + A_2^T \mathbf{y}_2 + \mathbf{s} - \mathbf{c} \right\|^2$$

Differentiating with respect to $\mathbf{y}_1$:

$$\nabla_{\mathbf{y}_1} L^d(\mathbf{y}, \mathbf{s}, \mathbf{x}) = -\mathbf{b}_1 - A_1 \mathbf{x} + \beta A_1 \left(A_1^T \mathbf{y}_1 + A_2^T \mathbf{y}_2 + \mathbf{s} - \mathbf{c} \right)$$

Setting the gradient to 0, we obtain the update for $\mathbf{y}_1$:

$$\begin{aligned} A_1 \left(A_1^T \mathbf{y}_1^{k+1} + A_2^T \mathbf{y}_2^k + \mathbf{s} - \mathbf{c} \right) &= \frac{1}{\beta} \left(A_1 \mathbf{x} + \mathbf{b}_1 \right) \\ A_1 A_1^T \mathbf{y}_1^{k+1} &= \frac{1}{\beta} \left(A_1 \mathbf{x} + \mathbf{b}_1 \right) - A_1 \left(A_2^T \mathbf{y}_2^k + \mathbf{s} - \mathbf{c} \right) \\ \mathbf{y}_1^{k+1} &= \left(A_1 A_1^T \right)^{-1} \left(\frac{1}{\beta} \left(A_1 \mathbf{x} + \mathbf{b}_1 \right) - A_1 \left(A_2^T \mathbf{y}_2^k + \mathbf{s} - \mathbf{c} \right) \right) \end{aligned}$$

By symmetry, the update for $\mathbf{y}_2$ is:

$$\mathbf{y}_2^{k+1} = \left(A_2 A_2^T \right)^{-1} \left(\frac{1}{\beta} \left(A_2 \mathbf{x} + \mathbf{b}_2 \right) - A_2 \left(A_1^T \mathbf{y}_1^{k+1} + \mathbf{s} - \mathbf{c} \right) \right)$$

assuming that we update $\mathbf{y}_2$ after $\mathbf{y}_1$. As in the primal case, we may need to randomly permute the update order at each iteration. Note: we can easily extend the above result to a general number of blocks.

Using Preconditioning for Dual Block Splitting

When preconditioning is applied to the dual LP problem (whether or not implemented with the interior point method), the variables $\mathbf{y}$ become separable. To see this, consider (22), where the coefficient of $\mathbf{y}$ is AA^T. When standard preconditioning is applied, we have

$$A'(A')^T = (AA^T)^{-1/2} AA^T (AA^T)^{-1/2} = (AA^T)^{-1/2} (AA^T)^{1/2} (AA^T)^{1/2} (AA^T)^{-1/2} = I.$$

Therefore, the variables in $\mathbf{y}$ become separable and block-splitting produces the same result as non-block-splitting. This explains the dual Experiments, where the number of iterations required by the dual is independent of the number of blocks when preconditioning is applied.

Derivation of Updates for ADMM SDP Primal

$$\nabla_{X_1} L = C - \mathcal{A}^T y - Z + \beta \mathcal{A}^T \left(\mathcal{A} X_1 - b \right) + \beta \left(X_1 - X_2 \right)$$

where

$$\mathcal{A}^T y = \sum_{i=1}^{m} y_i A_i$$

1st order condition for X_1:

$$C - \mathcal{A}^T y - Z + \beta \mathcal{A}^T \left(\mathcal{A} X_1 - b \right) + \beta \left(X_1 - X_2 \right) = 0$$

X_1 update:

$$X_1 = \left(\mathcal{A}^T \mathcal{A} + I \right)^{-1} \left(X_2 + \frac{1}{\beta} \left(\mathcal{A}^T y + Z - C \right) + \mathcal{A}^T b \right)$$

1st order condition for X_2:

$$\nabla_{X_2} L = -Z - \beta \left(X_1 - X_2 \right)$$

The X_2 update uses this 1st order condition and enforces the PSD requirement:

$$X_2 = \max\{0, -\frac{Z}{\beta} - X_1\}$$

Here, the operation $\max\{0, M\}$ means to set any negative eigenvalues of M to 0. In particular, consider the eigendecomposition of M: $M = VDV^{-1}$. We compute $\hat{D} = max\{0, D\}$ (this max is computed element-wise), then set $X_2 = V\hat{D}V^{-1}$.

Derivation of Updates for ADMM SDP Dual

The 1st order gradient condition for y

$$\nabla_y L = \mathbf{b} - \mathcal{A}X + \beta\mathcal{A}\left(\mathcal{A}^T\mathbf{y} + S - C\right) = \mathbf{0}$$

leads to the y update:

$$y = \left(\mathcal{A}\mathcal{A}^T\right)^{-1}\left(\frac{1}{\beta}\left(\mathcal{A}X - b\right) + \mathcal{A}\left(C - S\right)\right)$$

The 1st order condition for S

$$\nabla_S L = -X + \beta\left(\mathcal{A}^T\mathbf{y} + S - C\right) = \mathbf{0}$$

leads to the S update (which must enforce the PSD constraint):

$$S = \max\left\{0, \frac{1}{\beta}X + C - \mathcal{A}^T y\right\}$$