Multi-block ADMM Methods for Linear Programming

Nico Chaves, Junjie (Jason) Zhu

December 11, 2016

1 Introduction

The alternating direction method of multipliers (ADMM) is a first-order method that can be used to solve various optimization problems. It splits the original problem into smaller pieces that are easier to handle during each iterative update. ADMM relies on the idea of the augmented Lagrangian method to robustify the problem and the splitting procedure makes it a proximal method that can be adopted in a wide range of optimization problems.

The goal of this project is to use ADMM to solve the primal linear programming (LP) problem:

$$\begin{aligned} \text{minimize}_{\mathbf{x}} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0} \end{aligned} \tag{OPT1}$$

or its dual

$$\begin{aligned} \text{maximize}_{\mathbf{y},\mathbf{s}} \quad & \mathbf{b}^T\mathbf{y} \\ \text{subject to} \quad & A^T\mathbf{y} + \mathbf{s} = \mathbf{c}, \\ & \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{OPT2}$$

In this report, we first re-formulate the problems to solve for closed form updates on the primal and the dual LP problems. We then approach each problem using the corresponding interior-point method. For each of these four approaches, we are specifically interested in the effect of preconditioning and block-splitting on the convergence rate and the accuracy of the solvers. We implemented all of the algorithms in MATLAB and ran experiments to test how these methods perform on different simulated problems.

Preconditioning

In this report, we demonstrate the effect of pre-conditioning on the various ADMM approaches mentioned above. To pre-condition a linear programming problem, one first computes

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

and then substitutes A' and $\mathbf{b}'$ for A and $\mathbf{b}$ in the original problems (OPT1) and (OPT2). Clearly, this re-formulation does not change the feasible regions for (OPT1) or (OPT2).

Block-Splitting ADMM

As we will show in Section 2, ADMM requires computation of a matrix inverse. Since matrix inversion is approximately cubic in complexity, this computation may slow down the algorithm 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.

In this report, we demonstrate the effect of block splitting on ADMM. We demonstrate that block splitting with 3 or more blocks does not necessarily lead to a convergent solution. However, we demonstrate through experiments that randomly permuting the update order of the blocks at each iteration resolves this issue.

We also demonstrate that block splitting and preconditioning work well when used together. However, the preconditioning approach described above requires a matrix inverse computation. Therefore, one still may suffer from a slow matrix inversion if one uses preconditioning. However, one may use an inverse-free method such as Cholesky preconditioning. We plan to study this procedure in a future work.

2 Basic ADMM for LP

Primal Re-formulation

We re-formulate the primal LP problem as:

$$\begin{aligned} \text{minimize}_{\mathbf{x}_1,\mathbf{x}_2} \quad & \mathbf{c}^T\mathbf{x}_1 \\ \text{subject to} \quad & A\mathbf{x}_1 = \mathbf{b} \\ & \mathbf{x}_1 - \mathbf{x}_2 = \mathbf{0} \\ & \mathbf{x}_2 \geq \mathbf{0} \end{aligned} \tag{OPT3}$$

and consider the split Lagrangian function:

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

Primal Update

To obtain the updates for the primal variables in ADMM, we must compute the gradient of these variables and set them to zero. Here we update $\mathbf{x}_1$ and $\mathbf{x}_2$ sequentially using the update function for each respectively:

$$\mathbf{x}_1^{k+1} = \arg\min_{\mathbf{x}_1^k} L^P(\mathbf{x}_1^k,\mathbf{x}_2^k,\mathbf{y}^k,\mathbf{s}^k), \tag{2}$$

$$\mathbf{x}_2^{k+1} = \arg\min_{\mathbf{x}_2^k \geq 0} L^P(\mathbf{x}_1^{k+1},\mathbf{x}_2^k,\mathbf{y}^k,\mathbf{s}^k). \tag{3}$$

To solve for $\mathbf{x}_1$, we set the gradient to zero:

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

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

$$\mathbf{x}_1^{k+1} = \left(A^TA + I\right)^{-1}\left(\frac{1}{\beta}A^T\mathbf{y}^k + \frac{1}{\beta}\mathbf{s}^k - \frac{1}{\beta}\mathbf{c} + A^T\mathbf{b}^k + \mathbf{x}_2^k\right) \tag{4}$$

Similarly, we solve the gradient

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

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:

$$x_{2,j}^{k+1} = \max\left\{x_{1,j}^k - \frac{1}{\beta}s_j^k, 0\right\} \text{ for } j = 1,..,n. \tag{5}$$

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

$$\mathbf{y}^{k+1} = \mathbf{y}^k + \beta(A\mathbf{x}_1^{k+1} - \mathbf{b}) \tag{6}$$

$$\mathbf{s}^{k+1} = \mathbf{s}^k - \beta(\mathbf{x}_1^{k+1} - \mathbf{x}_2^{k+1}) \tag{7}$$

These two updates guarantee that $\left(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}, \mathbf{y}^{k+1}, \mathbf{s}^{k+1}\right)$ is dual feasible, i.e.,

$$\nabla_{\mathbf{x}_1, \mathbf{x}_2} L^P(\mathbf{x}_1^k, \mathbf{x}_2^k, \mathbf{y}^k, \mathbf{s}^k) = \mathbf{0}.$$

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 Re-formulation

We convert the problem of the dual to a minimization problem to apply ADMM:

$$\begin{aligned} \text{minimize}_{\mathbf{y},\mathbf{s}} \quad & -\mathbf{b}^T\mathbf{y} \\ \text{subject to} \quad & A^T\mathbf{y} + \mathbf{s} = \mathbf{c}, \\ & \mathbf{s} \geq \mathbf{0}. \end{aligned} \tag{OPT4}$$

The dual LP problem does not require additional re-formulation because the non-zero variable $\mathbf{s}$ is already separable, which can be observed via the dual Lagrangian function:

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

Note that $\mathbf{x}$ would be non-positive since we changed maximization to minimization and we would need to take the negative of $\mathbf{x}$ of the final solution as the solution to the original LP.

Dual Update

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

$$\mathbf{y}^{k+1} = \arg\min_{\mathbf{y}} L^d(\mathbf{y}^k, \mathbf{s}^k, \mathbf{x}^k), \tag{8}$$

$$\mathbf{s}^{k+1} = \arg\min_{\mathbf{s}\geq 0} L^d(\mathbf{y}^{k+1}, \mathbf{s}^k, \mathbf{x}^k). \tag{9}$$

Setting the gradients with respect to $\mathbf{y}$ and $\mathbf{s}$ to zero, i.e.,

$$\nabla_{\mathbf{y}} L^d(\mathbf{y}, \mathbf{s}, \mathbf{x}) = -\mathbf{b} - A\mathbf{x} + \beta A\left(A^T\mathbf{y} + \mathbf{s} - \mathbf{c}\right) = \mathbf{0}, \tag{10}$$

$$\nabla_{\mathbf{s}} L^d(\mathbf{y}, \mathbf{s}, \mathbf{x}) = -\mathbf{x} + \beta\left(A^T\mathbf{y} + \mathbf{s} - \mathbf{c}\right) = \mathbf{0}, \tag{11}$$

we have the updates for $\mathbf{y}$ (least-squares problem) and $\mathbf{s}$:

$$\mathbf{y}^{k+1} = \left(AA^T\right)^{-1}\left(\frac{1}{\beta}\left(A\mathbf{x}^k + \mathbf{b}\right) - A\mathbf{s}^k + A\mathbf{c}\right), \tag{12}$$

$$s_j^{k+1} = \max\left\{\frac{1}{\beta}x_j^k - (A^T\mathbf{y}^{k+1})_j + c_j, 0\right\}. \tag{13}$$

We can update the multipliers $\mathbf{x}$ by

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \beta\left(A^T\mathbf{y}^{k+1} + \mathbf{s}^{k+1} - \mathbf{c}\right). \tag{14}$$

The termination condition for this dual would be $A\mathbf{x}^{k+1} + \mathbf{b} \approx \mathbf{0}$.

3 Interior-Point ADMM for LP

Primal LP Updates

We can use the previous formulation in (OPT3) with $\mathbf{x}_1$ and $\mathbf{x}_2$ using the barrier function:

$$\begin{aligned} \text{minimize}_{\mathbf{x}_1,\mathbf{x}_2} \quad & \mathbf{c}^T\mathbf{x}_1 + \mu \sum_j \ln((x_{2,j}) \\ \text{subject to} \quad & A\mathbf{x}_1 = \mathbf{b}, \\ & \mathbf{x}_1 - \mathbf{x}_2 = \mathbf{0}, \\ & \mathbf{x}_2 > \mathbf{0}, \end{aligned} \tag{OPT5}$$

where the Lagrangian function is given by

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

Setting the gradient of this to zero yields the same update for $\mathbf{x}_1$ as in (4). However, the we need to modify the update for $\mathbf{x}_2$ as now we have:

$$s_j - \frac{\mu}{x_{2,j}} + \beta(x_{2,j} - x_{1,j}) = 0 \tag{15}$$

which yields

$$x_{2,j} = \frac{1}{2\beta}\left(\beta x_{1,j} - s_j + \sqrt{\beta^2 x_{1,j}^2 - 2\beta s_j x_{1,j} + s_j^2 + 4\beta\mu}\right), \text{ for } j = 1,..,n \tag{16}$$

The update equations for $\mathbf{y}$ and $\mathbf{s}$ remain the same as in (6) and (7). The update equation for $\mathbf{x}$ remains the same as in (14). At the end of each iteration, we update $\mu^{k+1} = \gamma\mu^k$ for some constant $\gamma < 1$, and the terminal condition remains the same as the non-interior-point method.

Dual LP Updates

We can use the previous formulation in (OPT4) using the barrier function:

$$\begin{aligned} \text{minmize}_{\mathbf{y},\mathbf{s}} \quad & -\mathbf{b}^T\mathbf{y} + \mu \sum_j \ln(s_j) \\ \text{subject to} \quad & A^T\mathbf{y} + \mathbf{s} = \mathbf{c}, \\ & \mathbf{s} > \mathbf{0}. \end{aligned} \tag{OPT6}$$

where the Lagrangian function is given by

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

Setting the gradient of this to zero yields the same update for $\mathbf{y}$ as in (12) However, the we need to modify the update for $\mathbf{s}$ as now we have:

$$-\frac{\mu}{s_j} + x_j + \beta\left(s_j - (\mathbf{c} - (A^T\mathbf{y}))_j\right) = 0 \tag{17}$$

which yields

$$s_j = \frac{1}{2\beta}\left(\beta(\mathbf{c} - (A^T\mathbf{y}))_j + x_j + \sqrt{\beta^2(\mathbf{c} - (A^T\mathbf{y}))_j^2 - 2\beta(\mathbf{c} - (A^T\mathbf{y}))_j x_j + x_j^2 + 4\beta\mu}\right), \text{ for } j = 1,..,n \tag{18}$$

The update equation for $\mathbf{x}$ remain the same as in (14). At the end of each iteration, we update $\mu^{k+1} = \gamma\mu^k$ for some constant $\gamma < 1$, and the terminal condition remains the same as the non-interior-point method.

Multi-Block ADMM

Primal with Block Splitting

For simplicity, we first consider splitting the problem into 2 blocks of equal size as follows:

$$\mathbf{x}_1 = \begin{bmatrix} \mathbf{x}_{1,1} \\ \mathbf{x}_{1,2} \end{bmatrix}$$

$$A = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$$

Then, the Lagrangian can be expressed as:

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

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 $\mathbf{x}_{1,2}$. If we use a randomized update order, then we may end up updating $\mathbf{x}_{1,2}$ first instead.

Block Splitting (for a general number of blocks)

We can easily extend the above result to a general number of blocks. 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:

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

where A_i refers to the i^{th} block of columns of A.

Dual ADMM

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

Dual ADMM With Block Splitting

For simplicity, we first consider splitting the problem into 2 blocks of equal size as follows:

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

$$A^T = \begin{bmatrix} A_1^T & A_2^T \end{bmatrix}$$

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 $\mathbf{y}_1$:

$$\begin{aligned} A_1\left(A_1^T\mathbf{y}_1^{k+1} + A_2^T\mathbf{y}_2^k + \mathbf{s} - \mathbf{c}\right) &= \frac{1}{\beta}\left(A_1\mathbf{x} + \mathbf{b}_1\right) \\ A_1A_1^T\mathbf{y}_1^{k+1} &= \frac{1}{\beta}\left(A_1\mathbf{x} + \mathbf{b}_1\right) - A_1\left(A_2^T\mathbf{y}_2^k + \mathbf{s} - \mathbf{c}\right) \\ \mathbf{y}_1^{k+1} &= \left(A_1A_1^T\right)^{-1}\left(\frac{1}{\beta}\left(A_1\mathbf{x} + \mathbf{b}_1\right) - A_1\left(A_2^T\mathbf{y}_2^k + \mathbf{s} - \mathbf{c}\right)\right) \end{aligned}$$

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

$$\mathbf{y}_2^{k+1} = \left(A_2A_2^T\right)^{-1}\left(\frac{1}{\beta}\left(A_2\mathbf{x} + \mathbf{b}_2\right) - A_2\left(A_1^T\mathbf{y}_1^{k+1} + \mathbf{s} - \mathbf{c}\right)\right)$$

assuming that we update $\mathbf{y}_2$ after $\mathbf{y}_1$.

Dual Block Splitting (for a general number of blocks)

We can easily extend the above result to a general number of blocks. Let U denote the set of blocks which have already been updated at the current iteration. To update block i of $\mathbf{y}$:

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

Block-Splitting and Preconditioning

As discussed earlier, it may be useful to split variables (such as the primal variables for the primal LP) into blocks to avoid computing the inverse of a large matrix. However, splitting a set of variables into blocks is helpful only when these variables are *not* separable. Otherwise, updating variables a block simultaneously would yield the same result as updating the variables in the block one at at time, which means updating block by block would yield the same result as updating all the variables one at a time. At the end of each iteration, if these variables are separable, they would take identical values no matter how they are grouped or updated. Therefore, it only makes sense to split $\mathbf{x_1}$ in the primal problems (i.e., (OPT3) and (OPT5)), or to split $\mathbf{y}$ in the primal problems (i.e., (OPT4) and (OPT6)).

A sufficient condition for the variables to be separable is when the coefficient of these variables the gradient function is the identify matrix. Interestingly, we show that 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 the update equation (10), where the coefficient of $\mathbf{y}$ is AA^T. When pre-conditioning is applied, we have

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

4 Experiments and Results

We evaluated the performance of the different implementations of ADMM LP solvers by randomly simulating LP problems and recording the number of steps needed to converge. In particular, we simulated LP problems given by (OPT1) by randomly selecting $A \in \mathbb{R}^{mxn}$ and $c \in \mathbb{R}^n$. We only considered feasible problems. To ensure feasibility, we generated a random feasible point $\mathbf{x}_0 \geq 0$ and let $\mathbf{b} = A\mathbf{x}_0$. We considered a solution to be optimal when $|A\mathbf{x} - \mathbf{b}| < 10^{-3}$ (for the primal formulation) or $|A\mathbf{x} + \mathbf{b}| < 10^{-3}$ (for the dual formulation). Each set of experiments is described in more detail below.

Preconditioning

We first evaluated the effect of preconditioning on ADMM LP solvers that do not perform splitting on variables, and varied the tuning parameter β in the augmented Lagrangian function. NICO: please fill in a figure and some description.

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

First, we evaluated the ADMM solvers using sequential updating (i.e. we did not randomly permute the block update order) and without preconditioning. The results of this experiment are shown in Figure 4. 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 for the primal. As we will see in the following experiments, we can improve performance by introducing random permutations or preconditioning.

Block-splitting with Random Permutation (No Preconditioning)

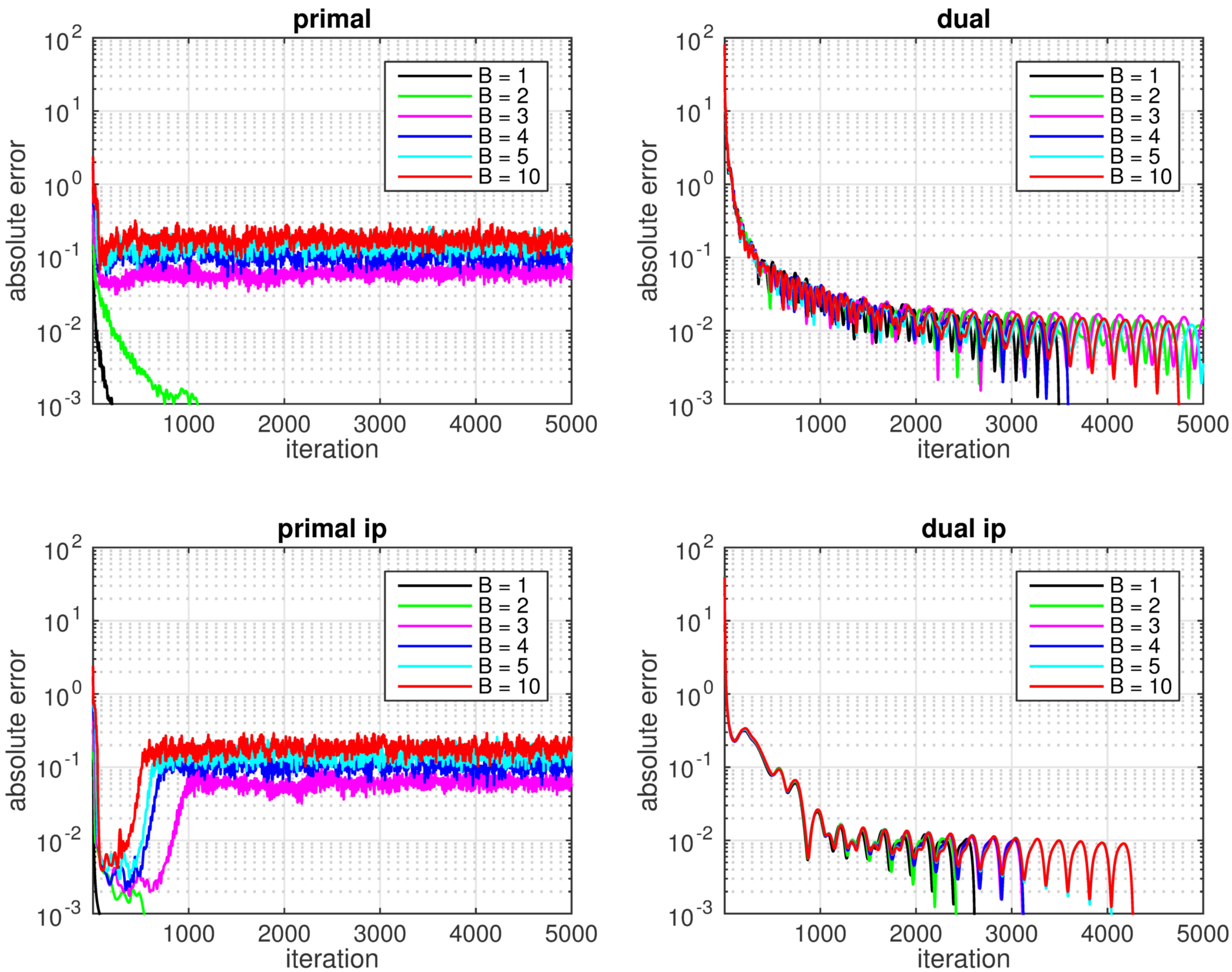
In this set of experiments, we randomly permuted the update order of the block variables at each iteration. The results are shown in Figure 2. When using random permutation, the primal now converges for all block sizes. Note that splitting the primal into more blocks requires more iterations. However, performing block splitting may still result in faster overall computation because of the faster inverse computations.

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 3. 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 1: No Preconditioning, No Random Permutation



Structured Problems

Analysis of Non-Converging Primal

In Figure , we observed that the absolute error oscillates for

5 Conclusions and Future Work

Figure 2: Random Permutation, No Preconditioning

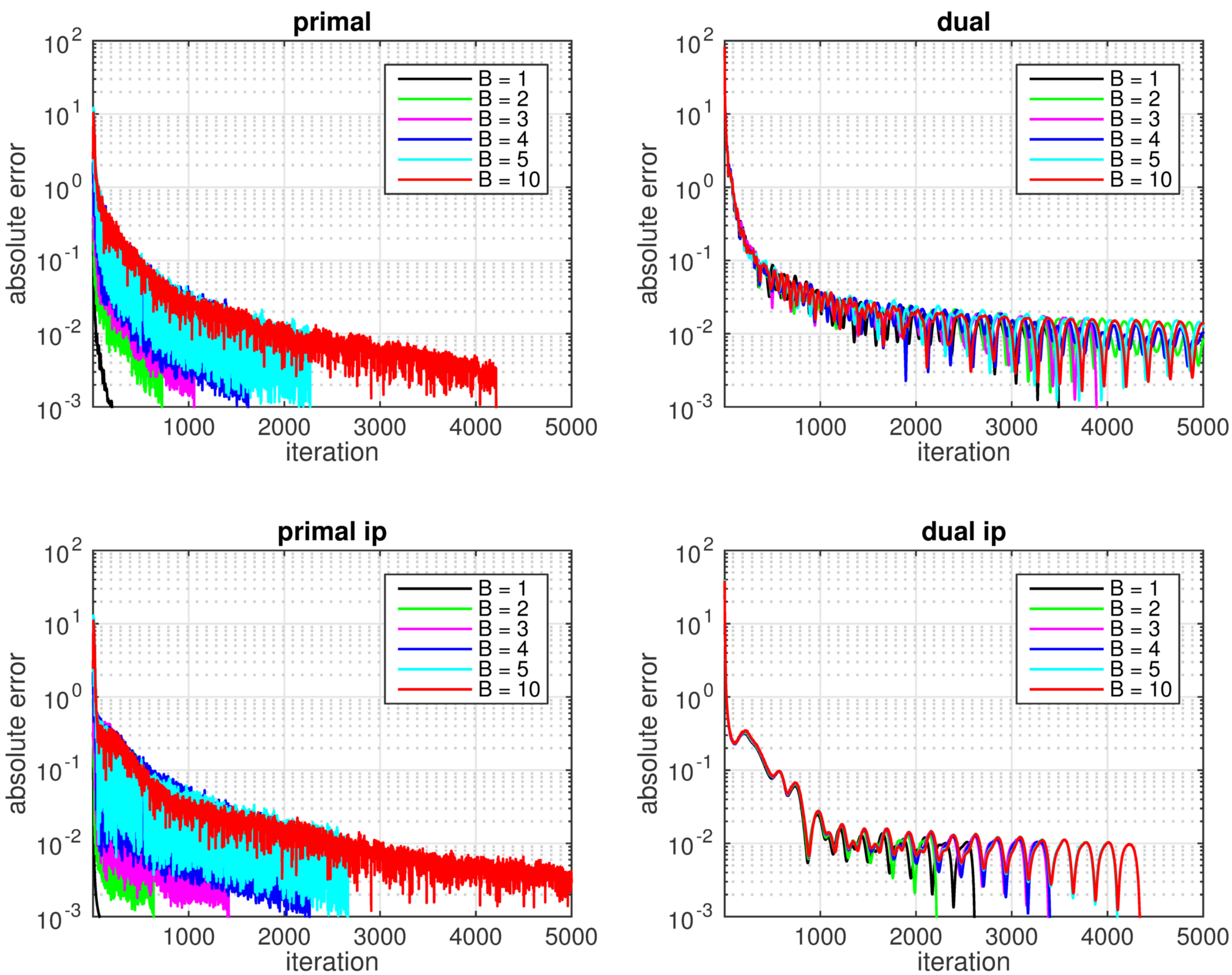


Figure 3: Preconditioning, No Random Permutation