MS&E 310 Course Project
ADMM for Linear Programming

Yuxin Ji
Qi Qin

December 8, 2017

Contents

1 Algorithms

1.1 Basic Primal ADMM: Two Blocks

Step 1: Update variable $\mathbf{x}_1$

$$\begin{aligned}
\mathbf{x}_1^{k+1} &= \underset{\mathbf{x_1}}{\operatorname{argmin}} L^p(\mathbf{x}_1, \mathbf{x}_2^k, \mathbf{y}^k, , \mathbf{s}^k) \\
&= \underset{\mathbf{x_1}}{\operatorname{argmin}} \mathbf{c}^T\mathbf{x}_1 - \mathbf{y}^{k^T}(\mathbf{A}\mathbf{x}_1 - \mathbf{b}) - \mathbf{s}^T(\mathbf{x}_1 - \mathbf{x}_2^k) + \frac{2}{\beta}(\|\mathbf{A}\mathbf{x}_1 - \mathbf{b}\|^2 + \|\mathbf{x}_1 - \mathbf{x}_2^k\|^2) \\
&= \underset{\mathbf{x}_1}{\operatorname{argmin}} \frac{\beta}{2}[\mathbf{x}_1^T(\mathbf{A}^T\mathbf{A} + \mathbf{I})\mathbf{x}_1 + (\frac{2}{\beta}\mathbf{c}^T - \frac{2}{\beta}\mathbf{y}^{k^T}\mathbf{A} - \frac{2}{\beta}\mathbf{s}^T - 2\mathbf{b}^T\mathbf{A} - 2\mathbf{x}_2^{k^T})\mathbf{x}_1] \\
&= (\mathbf{A}^T\mathbf{A} + \mathbf{I})^{-1}(-\frac{1}{\beta}\mathbf{c} + \frac{1}{\beta}\mathbf{A}^T\mathbf{y}^k + \frac{1}{\beta}\mathbf{s}^k + \mathbf{A}^T\mathbf{b} + \mathbf{x}_2^k).
\end{aligned}$$

Step 2: Update variable $\mathbf{x}_2$

$$\begin{aligned}
\mathbf{x}_2^{k+1} &= \underset{\mathbf{x_2}\geqslant 0}{\operatorname{argmin}} L^p(\mathbf{x}_1^{k+1}, \mathbf{x}_2, \mathbf{y}^k, , \mathbf{s}^k) \\
&= \underset{\mathbf{x_2}\geqslant 0}{\operatorname{argmin}} \mathbf{s}^{k^T}\mathbf{x}_2 + \frac{\beta}{2}\|\mathbf{x}_1^{k+1} - \mathbf{x}_2\|^2 \\
&= \underset{\mathbf{x_2}\geqslant 0}{\operatorname{argmin}} \frac{\beta}{2}[\mathbf{x}_2^T\mathbf{x}_2 - 2\mathbf{x}_1^{k+1}\mathbf{x}_2 + \mathbf{x}_1^{k^T}\mathbf{x}_1^{k+1} + \frac{2}{\beta}\mathbf{s}^T\mathbf{x}_2] \\
&= \underset{\mathbf{x_2}\geqslant 0}{\operatorname{argmin}} \mathbf{x}_2^T\mathbf{x}_2 + \frac{2}{\beta}\mathbf{s}^T\mathbf{x}_2 - 2\mathbf{x}_1^{k^T}\mathbf{x}_1^{k+1} \\
&= \underset{\mathbf{x_2}\geqslant 0}{\operatorname{argmin}} \sum_i [x_{2i}^2 - (2x_{1i}^{k+1} - \frac{2}{\beta}s_i^k)x_{2i}].
\end{aligned}$$

Thus

$$\mathbf{x}_2 = \max\{\mathbf{x}_1^{k+1} - \frac{1}{\beta}\mathbf{s}^k, \mathbf{0}\}.$$

Step 3: Update y and s

$$\begin{aligned}
\mathbf{y}^{k+1} &= \mathbf{y}^k - \beta(A\mathbf{x}_1^{k+1} - \mathbf{b}), \\
\mathbf{s}^{k+1} &= \mathbf{s}^k - \beta(\mathbf{x}_1^{k+1} - \mathbf{x}_2^{k+1}).
\end{aligned}$$

The way to get preconditioned form is similar.

1.2 Basic Dual ADMM

Step 1: Update variable y

$$\begin{aligned}
\mathbf{y}^{k+1} &= \underset{\mathbf{y}}{\operatorname{argmin}} L^d(\mathbf{y}, \mathbf{s}^k, \mathbf{x}^k) \\
&= \underset{\mathbf{y}}{\operatorname{argmin}} -\mathbf{b}^T\mathbf{y} - \mathbf{x}^{k^T}(\mathbf{A}^T\mathbf{y} + \mathbf{s}^k - \mathbf{c}) + \frac{\beta}{2}\|\mathbf{A}^T\mathbf{y} + \mathbf{s}^k - \mathbf{c}\|^2 \\
&= \underset{\mathbf{y}}{\operatorname{argmin}} \frac{\beta}{2}[\mathbf{y}^T\mathbf{A}\mathbf{A}^T\mathbf{y} + 2(\mathbf{s}^k - \mathbf{c})^T\mathbf{A}^T\mathbf{y} - \frac{2}{\beta}\mathbf{b}^T\mathbf{y} - \frac{2}{\beta}\mathbf{x}^{k^T}\mathbf{A}^T\mathbf{y}] \\
&= \underset{\mathbf{y}}{\operatorname{argmin}} \mathbf{y}^T\mathbf{A}\mathbf{A}^T\mathbf{y} + (2\mathbf{s}^{k^T} - 2\mathbf{c}^T\mathbf{A}^T - \frac{2}{\beta}\mathbf{b}^T - \frac{2}{\beta}\mathbf{x}^{k^T}\mathbf{A}^T)\mathbf{y} \\
&= (\mathbf{A}\mathbf{A}^T)^{-1}(-\mathbf{A}\mathbf{s}^k + \mathbf{A}\mathbf{c} + \frac{1}{\beta}\mathbf{b} + \frac{1}{\beta}\mathbf{A}\mathbf{x}^k).
\end{aligned}$$

Step 2: Update slack variable s

$$\begin{aligned}
\mathbf{s}^{k+1} &= \underset{\mathbf{s}\geqslant 0}{\operatorname{argmin}} L^d(\mathbf{y}^{k+1}, \mathbf{s}, \mathbf{x}^k) \\
&= \underset{\mathbf{s}\geqslant 0}{\operatorname{argmin}} -\mathbf{x}^{k^T}\mathbf{s} + \frac{\beta}{2}[\mathbf{s}^T\mathbf{s} + 2(\mathbf{y}^{k+1^T}A - \mathbf{c}^T)\mathbf{s}] \\
&= \underset{\mathbf{s}\geqslant 0}{\operatorname{argmin}} \frac{\beta}{2}[\mathbf{s}^T\mathbf{s} + 2(\mathbf{y}^{k+1^T}A - \mathbf{c}^T - \frac{1}{\beta}\mathbf{x}^{k^T})\mathbf{s} \\
&= \underset{\mathbf{s}\geqslant 0}{\operatorname{argmin}} \mathbf{s}^T\mathbf{s} - 2(-\mathbf{y}^{k+1^T}A + \mathbf{c}^T + \frac{1}{\beta}\mathbf{x}^{k^T})\mathbf{s}.
\end{aligned}$$

We can then get $s_i^{k+1} = \begin{cases} (-A^T\mathbf{y}^{k+1})_i + c_i + \frac{1}{\beta}x_i^k, & (-A^T\mathbf{y}^{k+1})_i + c_i + \frac{1}{\beta}x_i^k \geqslant 0 \\ 0 & \text{otherwise.} \end{cases}$.

Thus the explicit form is

$$\mathbf{s}^{k+1} = \max\{-\mathbf{A}^T\mathbf{y}^{k+1} + \mathbf{c} + \frac{1}{\beta}\mathbf{x}^k, \mathbf{0}\}.$$

Step 3: Update x

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

1.3 Interior Point ADMM for Primal

$$\min \quad \mathbf{c}^T\mathbf{x}_1 - \mu\sum_{j=1}^{n}\log(x_{2j})$$

subject to

$$\begin{aligned}&\mathbf{A}\mathbf{x}_1 = \mathbf{b}\\&\mathbf{x}_1 - \mathbf{x}_2 = \mathbf{0}\\&\mathbf{x}_2 > 0.\end{aligned}$$

We write out its Lagrangian function:

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

Step 1: Update variable $\mathbf{x}_1$

$$\begin{aligned}\mathbf{x}_1^{k+1} &= \underset{\mathbf{x_1}}{\operatorname{argmin}} L_\mu^p(\mathbf{x}_1, \mathbf{x}_2^k, \mathbf{y}^k, \mathbf{s}^k)\\&= \underset{\mathbf{x_1}}{\operatorname{argmin}} \mathbf{c}^T\mathbf{x}_1 - \mathbf{y}^{k^T}(\mathbf{A}\mathbf{x}_1 - \mathbf{b}) - \mathbf{s}^{k^T}(\mathbf{x}_1 - \mathbf{x}_2^k) + \frac{2}{\beta}(\|\mathbf{A}\mathbf{x}_1 - \mathbf{b}\|^2 + \|\mathbf{x}_1 - \mathbf{x}_2^k\|^2)\\&= \underset{\mathbf{x}_1}{\operatorname{argmin}} \frac{\beta}{2}[\mathbf{x}_1^T(\mathbf{A}^T\mathbf{A} + \mathbf{I}) - 2(\mathbf{b}^T\mathbf{A} + \mathbf{x}_2^{k^T} - \frac{1}{\beta}\mathbf{c}^T + \frac{1}{\beta}\mathbf{y}^{k^T}\mathbf{A} + \frac{1}{\beta}\mathbf{s}^{k^T})\mathbf{x}_1]\\&= (\mathbf{A}^T\mathbf{A} + \mathbf{I})^{-1}(\mathbf{A}^T\mathbf{b} + \mathbf{x}_2^k - \frac{1}{\beta}\mathbf{c} + \frac{1}{\beta}\mathbf{A}^T\mathbf{y}^k + \frac{1}{\beta}\mathbf{s}^k)\end{aligned}$$

.

Step 2: Update variable $\mathbf{x}_2$

$$\begin{aligned}\mathbf{x}_2 &= \underset{\mathbf{x_2}}{\operatorname{argmin}} L_\mu^p(\mathbf{x}_1^{k+1}, \mathbf{x}_2, \mathbf{y}^k, , \mathbf{s}^k)\\&= \underset{\mathbf{x_2}}{\operatorname{argmin}} -\mu\sum_j \log(x_{2j}) + \mathbf{s}^{k^T}\mathbf{x}_2 + \frac{\beta}{2}\|\mathbf{x}_1^{k+1} - \mathbf{x}_2\|^2\end{aligned}$$

By first order condition:

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

Let $\mathbf{d} = \beta\mathbf{x}_1^{k+1} - \mathbf{s}^k$, then we can get

$$\mathbf{x}_2^{k+1} = \frac{1}{2\beta}[\mathbf{d} + \sqrt{\mathbf{d}_\cdot^2 + 4\beta\mu\mathbf{e}}]$$

where $\mathbf{d}_\cdot^2 = [d_1^2, d_2^2, ..., d_n^2]^T$, $\mathbf{e} = (1, 1, ..., 1)^T$.

Step 3: Update y and s
The update formula is the same as that in Algorithm 1.

1.4 Interior Point ADMM for Dual

$$\max \quad \mathbf{b}^T\mathbf{y} + \mu \sum_{j=1}^{n} \log(s_j)$$

subject to

$$\begin{aligned} &\mathbf{A}^T\mathbf{y} + \mathbf{s} = \mathbf{c} \\ &\mathbf{s} > 0. \end{aligned}$$

whereby the augmented Lagrangian equation:

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

Step 1: Update variable y

$$\begin{aligned} \mathbf{y}^{k+1} &= \underset{\mathbf{y}}{\operatorname{argmin}}\, L_\mu^d(\mathbf{y}, \mathbf{s}^k, \mathbf{x}^k) \\ &= \underset{\mathbf{y}}{\operatorname{argmin}} -\mathbf{b}^T\mathbf{y} - \mathbf{x}^{k^T}\mathbf{A}^T\mathbf{y} + \frac{\beta}{2}[\mathbf{y}^T\mathbf{A}\mathbf{A}^T\mathbf{y} + 2(\mathbf{s} - \mathbf{c})^T\mathbf{A}^T\mathbf{y}] \\ &= \underset{\mathbf{y}}{\operatorname{argmin}} \frac{\beta}{2}[\mathbf{y}^T\mathbf{A}\mathbf{A}^T\mathbf{y} + 2((\mathbf{s}^{k^T} - \mathbf{c}^T)\mathbf{A}^T - \frac{1}{\beta}\mathbf{b}^T - \frac{1}{\beta}\mathbf{x}^{k^T}\mathbf{A}^T)\mathbf{y}] \\ &= (\mathbf{A}\mathbf{A}^T)^{-1}[\frac{1}{\beta}\mathbf{b} + \frac{1}{\beta}\mathbf{A}\mathbf{x}^k + \mathbf{A}(\mathbf{c} - \mathbf{s}^k)] \end{aligned}$$

Step 2: Update slack variable s

$$\begin{aligned} \mathbf{s}^{k+1} &= \underset{\mathbf{s}\geqslant 0}{\operatorname{argmin}}\, L_\mu^d(\mathbf{y}^{k+1}, \mathbf{s}, \mathbf{x}^k) \\ &= \underset{\mathbf{s}\geqslant 0}{\operatorname{argmin}} -\mu \sum_j \log(s_j) - \mathbf{x}^{k^T}\mathbf{s} + \frac{\beta}{2}[\mathbf{s}^T\mathbf{s} + 2(\mathbf{A}^T\mathbf{y} - \mathbf{c})^T\mathbf{s}] \end{aligned}$$

By first order condition:

$$-\mu\mathbf{S}^{-1} + \beta\mathbf{s} + \beta(\mathbf{A}^T\mathbf{y}^{k+1} - \mathbf{c}) - \mathbf{x}^k = 0$$

Let $\mathbf{k} = \mathbf{x} - \beta(\mathbf{A}^T\mathbf{y} - \mathbf{c})$. Thus we can get:

$$\mathbf{s} = \frac{1}{2\beta}(\mathbf{k} + \sqrt{\mathbf{k}^2 + 4\beta\mu\mathbf{e}})$$

where $\mathbf{e} = (1, 1, ..., 1)^T$.

Step 3: Update x
The update formula is similar to that in Algorithm 2.

1.5 Multiblock ADMM for Primal

By splitting the problem into N blocks, we solve the following problem

$$\min \quad \sum_{i=1}^{N} \mathbf{c}_i^T \mathbf{x}_{1,i}$$

subject to

$$\begin{aligned}
&\sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}_{1,i} = \mathbf{b} \\
&\mathbf{x}_{1,i} - \mathbf{x}_{2,i} = 0, i = 1, ..., N, \\
&\mathbf{x}_{2,i} \geq 0, i = 1, ..., N
\end{aligned}$$

The corresponding Langrangian function is:

$$\begin{aligned}
&L^P(\mathbf{x}_{1,1}, ...\mathbf{x}_{1,N}, \mathbf{x}_{2,1}, ...\mathbf{x}_{2,N}, \mathbf{y}, \mathbf{s}_1, ..., \mathbf{s}_N) \\
&= \sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}_{1,i} - \mathbf{y}^T (\sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}_{1,i} - \mathbf{b}) - \sum_{i=1}^{N} \mathbf{s}_i^T (\mathbf{x}_{1,i} - \mathbf{x}_{2,i}) \\
&+ \frac{\beta}{2} (\| \sum_{i=1}^{N} \mathbf{A}_i \mathbf{x}_{1,i} - \mathbf{b} \|^2 + \sum_{i=1}^{N} \| \mathbf{x}_{1,i} - \mathbf{x}_{2,i} \|^2)
\end{aligned}$$

We will update the variables in a fixed order and randomly permutate the update order.

For the fixed order,

Update $\mathbf{x}_{1,i}, i = 1...., N$

$$\begin{aligned}
\mathbf{x}_{1,i}^{k+1} &= \underset{\mathbf{x_{1,i}}}{\operatorname{argmin}} L^P(\mathbf{x}_{1,1}^{k+1}, ..., \mathbf{x}_{1,i-1}^{k+1}, \mathbf{x}_{1,i}, \mathbf{x}_{1,i+1}^{k} ..., \mathbf{x}_{1,N}^{k}, \mathbf{x}_{2,1}^{k}, ...\mathbf{x}_{2,N}^{k}, \mathbf{y}^k, \mathbf{s}_1^k, ..., \mathbf{s}_N^k) \\
&= \underset{\mathbf{x_{1,i}}}{\operatorname{argmin}} \mathbf{c}_i^T \mathbf{x}_{1,i} - \mathbf{y}^{kT} \mathbf{A}_i \mathbf{x}_{1,i} - \mathbf{s}_i^{kT} \mathbf{x}_{1,i} + \frac{\beta}{2} (\mathbf{x}_{1,i}^T \mathbf{A}_i^T \mathbf{A}_i \mathbf{x}_{1,i} + 2(\sum_{j<i} \mathbf{x}_{1,j}^{k+1T} \mathbf{A}_j^T \mathbf{A}_i \mathbf{x}_{1,i} + \\
&\sum_{j>i} \mathbf{x}_{1,j}^{kT} \mathbf{A}_j^T \mathbf{A}_i \mathbf{x}_{1,i} - \mathbf{b}^T \mathbf{A}_i \mathbf{x}_{1,i}) + \mathbf{x}_{1,i}^T \mathbf{x}_{1,i} - 2\mathbf{x}_{2,i}^{kT} \mathbf{x}_{1,i}) \\
&= \underset{\mathbf{x_{1,i}}}{\operatorname{argmin}} \frac{\beta}{2} [\mathbf{x}_{1,i}^T (\mathbf{A}_i^T \mathbf{A}_i + \mathbf{I}) \mathbf{x}_{1,i} - 2(\mathbf{b}^T \mathbf{A}_i + \mathbf{x}_{2,i}^{kT} - \frac{1}{\beta} \mathbf{c}_i^T + \frac{1}{\beta} \mathbf{y}^{kT} \mathbf{A}_i + \frac{1}{\beta} \mathbf{s}_i^{kT} + \\
&\sum_{j<i} \mathbf{x}_{1,j}^{k+1T} \mathbf{A}_j^T \mathbf{A}_i \mathbf{x}_{1,i} + \sum_{j>i} \mathbf{x}_{1,j}^{kT} \mathbf{A}_j^T \mathbf{A}_i \mathbf{x}_{1,i}] \\
&= (\mathbf{A}_i^T \mathbf{A}_i + \mathbf{I})^{-1} (\mathbf{A}_i^T \mathbf{b} + \sum_{j<i} \mathbf{x}_{1,j}^{k+1T} \mathbf{A}_j^T \mathbf{A}_i \mathbf{x}_{1,i} + \sum_{j>i} \mathbf{x}_{1,j}^{kT} \mathbf{A}_j^T \mathbf{A}_i \mathbf{x}_{1,i} + \mathbf{x}_{2,i}^k \\
&- \frac{1}{\beta} \mathbf{c}_i + \frac{1}{\beta} \mathbf{A}_i^T \mathbf{y}^k + \frac{1}{\beta} \mathbf{s}_i^k)
\end{aligned}$$

Update $\mathbf{x}_{2,i}, i = 1...., N$

$$\mathbf{x}_{2,i}^{k+1} = \underset{\mathbf{x_{2,i}}}{\operatorname{argmin}} L^P(\mathbf{x}_{1,1}^{k+1}, ..., \mathbf{x}_{1,N}^{k+1}, \mathbf{x}_{2,1}^{k+1}, ..., \mathbf{x}_{2,i-1}^{k+1}, \mathbf{x}_{2,i}, \mathbf{x}_{2,i+1}^{k}, ..., \mathbf{x}_{2,N}^{k}, \mathbf{y}^k, \mathbf{s}_1^k, ..., \mathbf{s}_N^k)$$
$$= \underset{\mathbf{x}_{2,i}}{\operatorname{argmin}} \mathbf{s}_i^{kT}\mathbf{x}_{2,i} + \frac{\beta}{2}(\mathbf{x}_{2,i}^T\mathbf{x}_{2,i} - 2\mathbf{x}_{1,i}^{k+1\,T}\mathbf{x}_{2,i} + \mathbf{x}_{1,i}^{k+1\,T}\mathbf{x}_{1,i}^{k+1})$$

The first order comndition implies:

$$\mathbf{x}_{2,i} = \max\{\mathbf{x}_{1,i}^{k+1} - \frac{1}{\beta}\mathbf{s}_i^k, \mathbf{0}\}.$$

For random permutation (RP-ADMM),

First we generate a random permutation σ of $\{1, 2, \ldots, 2N\}$. Then for $i = 1, 2, \ldots, 2N$, we update $\mathbf{x}_{1,\sigma i}$ if $\sigma(i) \leq N$; and update $\mathbf{x}_{2,\sigma(i-N)}$ if $\sigma(i) > N$.

The update formula is similar to that in the fixed order part,

$$\mathbf{x}_{1,i} = (A_i^TA_i + I)^{-1}(A_i^T\mathbf{b} + \sum_{\sigma^{-1}(j)<\sigma^{-1}(i)} \mathbf{x}_{1,j}^{k+1T}\mathbf{A}_j^T\mathbf{A}_i\mathbf{x}_{1,i} + \sum_{\sigma^{-1}(j)>\sigma^{-1}(i)} \mathbf{x}_{1,j}^{kT}\mathbf{A}_j^T\mathbf{A}_i\mathbf{x}_{1,i}$$
$$+ \mathbf{x}_{2,i}^k\mathbb{I}_{\sigma^{-1}(i+N)>\sigma^{-1}(i)} + \mathbf{x}_{2,i}^{k+1}\mathbb{I}_{\sigma^{-1}(i+N)<\sigma^{-1}(i)} - \frac{1}{\beta}\mathbf{c}_i + \frac{1}{\beta}\mathbf{A}_i^T\mathbf{y}^k + \frac{1}{\beta}\mathbf{s}_i^k).$$

$$\mathbf{x}_{2,i} = \max\{\mathbf{x}_{1,i}^k\mathbb{I}_{\sigma^{-1}(i)>\sigma^{-1}(i+N)} + \mathbf{x}_{1,i}^{k+1}\mathbb{I}_{\sigma^{-1}(i)<\sigma^{-1}(i+N)} - \frac{1}{\beta}\mathbf{s}_i^k, \mathbf{0}\}.$$

Then we update $\mathbf{y}$ and $\{\mathbf{s}_i\}_{i\leq N}$ using a similar method in Algorithm 1.

1.6 Multiblock ADMM for Dual

$$\max \quad \sum_{i=1}^N \mathbf{b}_i^T\mathbf{y}_i$$

subject to

$$\sum_{i=1}^N (\mathbf{A}^T)_i\mathbf{y}_i + \mathbf{s} = \mathbf{c}$$
$$\mathbf{s} \geq \mathbf{0}$$

The corresponding Lagrangian function is:

$$L^d(\mathbf{y}_1, ..., \mathbf{y}_N, \mathbf{s}, \mathbf{x}) = -\sum_{i=1}^N \mathbf{b}_i^T\mathbf{y}_i - \mathbf{x}^T(\sum_{i=1}^N (\mathbf{A}^T)_i\mathbf{y}_i + \mathbf{s} - \mathbf{c}) + \frac{\beta}{2}\|\sum_{i=1}^N (\mathbf{A}^T)_i\mathbf{y}_i + \mathbf{s} - \mathbf{c}\|^2$$

For the fixed order,

$$
\begin{aligned}
\mathbf{y}_i^{k+1} &= \underset{\mathbf{y}_i}{\operatorname{argmin}} L_\mu^d(\mathbf{y}_1^{k+1}, ..., \mathbf{y}_{i-1}^{k+1}, \mathbf{y}_i, \mathbf{y}_{i+1}^k, ..., \mathbf{y}_N^k, \mathbf{s}^k, \mathbf{x}^k) \\
&= \underset{\mathbf{y}_i}{\operatorname{argmin}} -\mathbf{b}_i^T \mathbf{y}_i - \mathbf{x}^{kT}(\mathbf{A}^T)_i \mathbf{y}_i + \frac{\beta}{2}[\mathbf{y}_i^T((\mathbf{A}^T)_i)^T(\mathbf{A}^T)_i)\mathbf{y}_i + \\
&2(\mathbf{s}^{kT} - \mathbf{c}^T + \sum_{j<i} \mathbf{y}_j^{k+1\,T}((\mathbf{A}^T)_j)^T + \sum_{j>i} \mathbf{y}_j^{k\,T}((\mathbf{A}^T)_j)^T)(\mathbf{A}^T)_i \mathbf{y}_i] \\
&= (((\mathbf{A}^T)_i)^T(\mathbf{A}^T)_i)^{-1}[\frac{1}{\beta}\mathbf{b}_i + \frac{1}{\beta}((\mathbf{A}^T)_i)^T \mathbf{x}^k \\
&+ ((\mathbf{A}^T)_i)^T(\mathbf{c} - \mathbf{s}^k - \sum_{j<i}(\mathbf{A}^T)_j \mathbf{y}_j^{k+1} - \sum_{j>i}(\mathbf{A}^T)_j \mathbf{y}_j^k)]
\end{aligned}
$$

For random permutation (RP-ADMM),

We generate a random permutation σ of $\{1, 2, \ldots, N\}$.

$$
\begin{aligned}
\mathbf{y}_i^{k+1} =& (((\mathbf{A}^T)_i)^T(\mathbf{A}^T)_i)^{-1}[\frac{1}{\beta}\mathbf{b}_i + \frac{1}{\beta}((\mathbf{A}^T)_i)^T \mathbf{x}^k \\
&+ ((\mathbf{A}^T)_i)^T(\mathbf{c} - \mathbf{s}^k - \sum_{\sigma^{-1}(j)<\sigma^{-1}(i)} (\mathbf{A}^T)_j \mathbf{y}_j^{k+1} - \sum_{\sigma^{-1}(j)>\sigma^{-1}(i)} (\mathbf{A}^T)_j \mathbf{y}_j^k)]
\end{aligned}
$$

Finally, update $\mathbf{s}$ and $\mathbf{x}$ using a similar approach in Algorithm 2.

2 Numerical Experiments

This section discusses the results and implications of numerical experiments using the six algorithms respectively, i.e. basic primal ADMM, basic Dual ADMM, interior point ADMM for the primal and dual LPs, and multiblock ADMM for primal and dual LPs.

The experiments are implemented on Matlab(R2017b), Intel Core i5 CPU @ 2.3 GHz processor, 8 GB RAM.

The constraint matrix $A_{m\times n}$ is randomly generated from the standard Gaussian distribution. We randomly draw initialization vectors $\mathbf{x}0$, $\mathbf{y}0$ and $\mathbf{s}0$ from the standard Gaussian distribution, and take the absolute value of $\mathbf{x}0$ and $\mathbf{s}0$, to compute $\mathbf{b}$ and $\mathbf{c}$ in order to ensure primal and dual feasibility

2.1 Basic Primal ADMM: Two Blocks

2.1.1 Problem setting

The problem starts with $\mathbf{x}_1^0 = \mathbf{1}$, $\mathbf{x}_2^0 = \mathbf{x}_1^0$, $\mathbf{y}^0 = \mathbf{0}$ and $\mathbf{s}^0 = \mathbf{1}$. The iteration will terminate at step k once the following condition is satisfied

$$\frac{||A\mathbf{x}_1^k - \mathbf{b}|| + ||\mathbf{x}_1^k - \mathbf{x}_2^k||}{1 + ||\mathbf{b}||} < tol. \tag{1}$$

Here we define the relative residue as the sum of norm of residues of the two constraints $A\mathbf{x}_1 - \mathbf{b} = \mathbf{0}$, $\mathbf{x}_1 - \mathbf{x}_2 = \mathbf{0}$, divided by the norm of $\mathbf{b}$. Hence Algorithm 1 will terminate once the relative residue is below the tolerance error level. For the rest of the numerical experiments, we set $tol = 1e^{-3}$. For the rest of this subsection, we take $\beta = 1.00$.

2.1.2 Sensitivity analysis of A

We first explore the performance of Algorithm 1 with different sizes of A. The convergence sensitivity results are summarized below in Figure 1 and Figure 2:

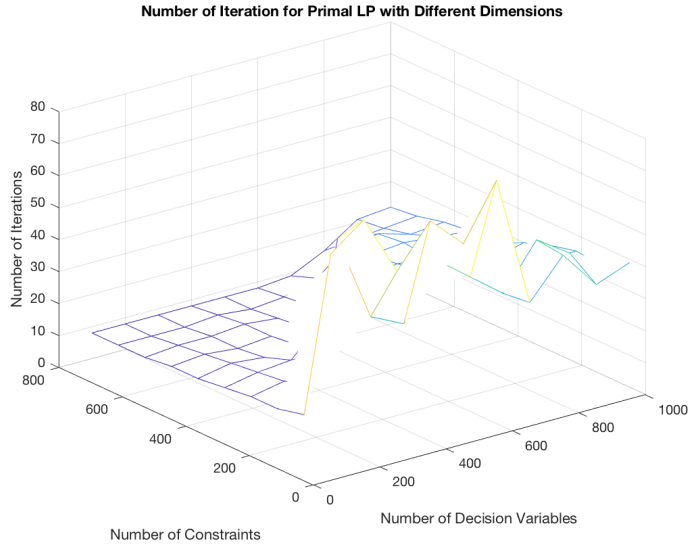

Fig. 1: Convergence of Steps with Different Sizes of A

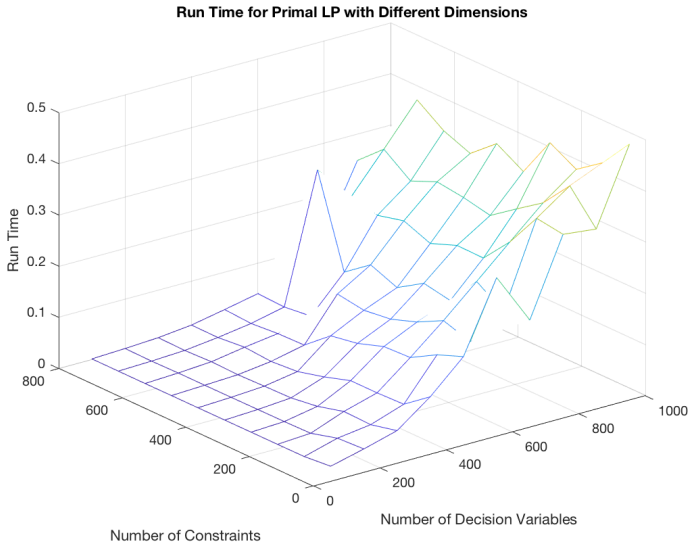

Fig. 2: Convergence of Run Time with Different Sizes of A

Figure 1 shows that Algorithm 1 converges in steps for $m \leq 800$ and $n \leq 1000$. Further, we observe the following three patterns:

- In general, the number of iterations increases in the number of decision variables (n).
- The number of iterations decreases slightly in the number of constraints m; and in particular, the convergence result is unstable with different n when m is small (in our experiment when $m < 400$).
- When $n < m$ and thus A has full column rank, Algorithm 1 converges with much fewer steps than when A has full row rank.

Figure 2 demonstrates similar convergence sensitivity results in run time:

- In general, run time increases in the number of decision variables (n).
- Run time decreases slightly in the number of constraints (m); it especially requires more time to converge with small number of constraint (in our experiment when $m < 400$); the convergence result with a small number of constraint is less unstable, compared to the convergence result in steps in Figure 1.
- When $n < m$ and thus A has full column rank, Algorithm 1 converges much faster than when A has full row rank.

Algorithm 1 does not perform well with small m, holding other conditions constant. Intuitively, by the stopping criterion, a single large residue $||(A\mathbf{x}_1 - \mathbf{b})_i|| + ||(\mathbf{x}_1 - \mathbf{x}_2)_i||$ would have a bigger impact on the overall relative residue when m is small than when m is large. Therefore, Algorithm 1 is unstable in terms of convergence steps when m is small. However, convergence in run time is not as much affected.

2.1.3 Convergence patterns

Since Algorithm 1 performs stably with large m and n, WLOG, we next take $A \in \mathbb{R}^{600 \times 1000}$ to explore the convergence results with basic ADMM and generalized ADMM, where the step-size to update $\mathbf{y}$ and $\mathbf{s}$ is $\alpha\beta$. Here we take $\alpha = 1.5$. The results are summarized in Figure 3 and Figure 4:

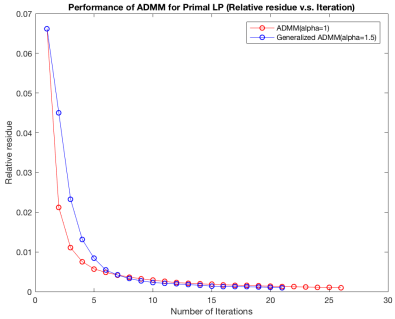

Fig. 3: Convergence of Steps(m=600, n=1000)

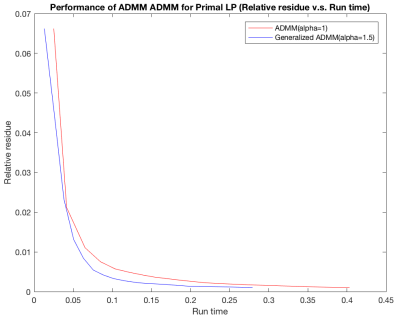

Fig. 4: Convergence of Time(m=600, n=1000)

Under both algorithms, the relative residue decreases fast at the begining and then slowly converges. We observe that generalized ADMM converges faster than basic ADMM both in steps and in run time. In particular, the relative residue of generalized ADMM algorithm first decreases more slowly than basic ADMM in steps, and then faster.

2.1.4 Sparsity

Finally, we explore the performance of Algorithm 1 under sparse $\mathbf{b}$ and $\mathbf{c}$. The sparse initialization vectors $\mathbf{x}0$ and $\mathbf{s}0$ are generated with randomly $n/5$ nonzeros drawn from the standard normal distribution, and $\mathbf{y}0$ with randomly $m/5$ nonzeros drawn from the standard normal distribution. We perform the algorithm on $A \in \mathbb{R}^{60\times 80}$ and $A \in \mathbb{R}^{3000\times 4000}$. Results are summarized in Figure 5 to Figure 12.

Although Algorithm 1 does not perform well for $A \in \mathbb{R}^{60\times 80}$, it converges much faster under sparsity. We further similuated 10 times, to find that the results are also much stabler than the original. On the other hand, the effect of sparsity is less significant for large constraint matrix $A \in \mathbb{R}^{3000\times 4000}$.

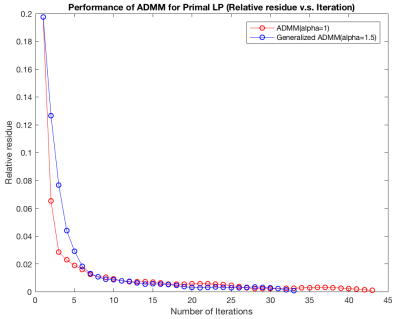

Fig. 5: Steps(m=60, n=80)

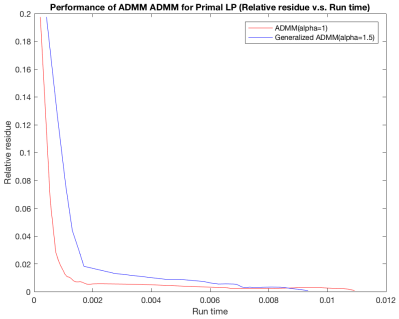

Fig. 6: Time(m=60, n=80)

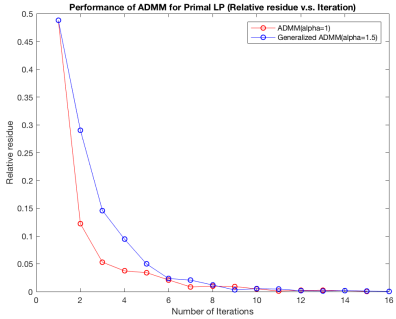

Fig. 7: Steps-Sparse (m=60, n=80)

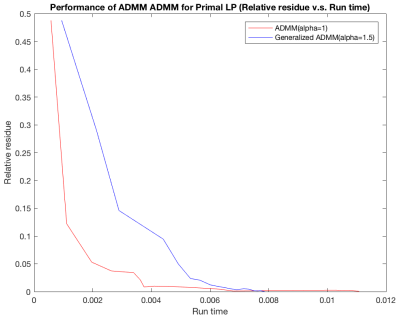

Fig. 8: Time-Sparse (m=60, n=80)

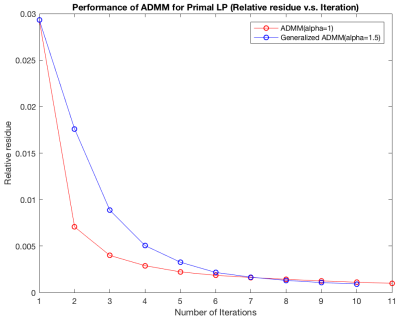

Fig. 9: Steps(m=3000, n=4000)

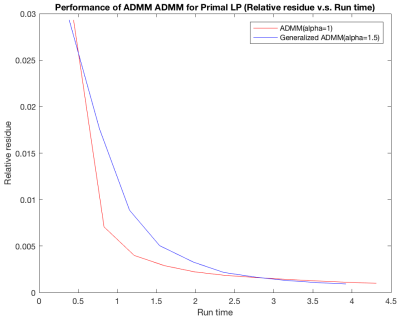

Fig. 10: Time(m=3000, n=4000)

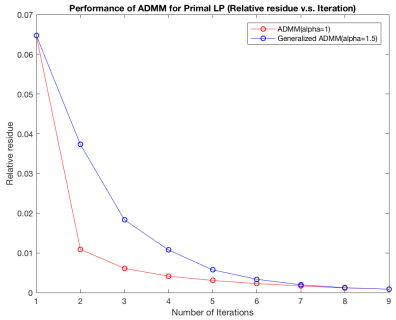

Fig. 11: Steps-Sparse(m=3000, n=4000)

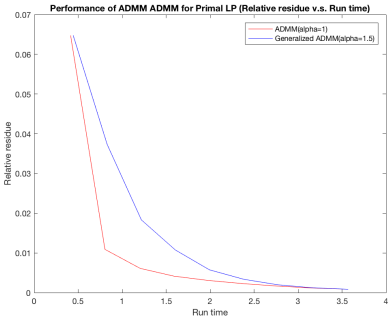

Fig. 12: Time-Sparse(m=3000, n=4000)

2.2 Basic Dual ADMM

2.2.1 Problem setting

The problem starts with $\mathbf{y}^0 = \mathbf{0}$, $\mathbf{s}^0 = \mathbf{1}$ and $\mathbf{x}^0 = \mathbf{1}$. The iteration will terminate at step k once the following condition is satisfied

$$\frac{||A^T\mathbf{y}^k + \mathbf{s}^k - \mathbf{c}||}{1 + ||\mathbf{c}||} < tol. \tag{2}$$

Using a similar definition in Algorithm 1, Algorithm 2 will terminate once the relative residue is below the tolerance error level. For the rest of this subsection, we take $\beta = 1.00$.

2.2.2 Sensitivity analysis of A

By the formula to update $\mathbf{y}$, A must be nonsingular, and thus $m \leq n$. The performance of Algorithm 2 with different sizes of A is summarized in Figure 13 and Figure 14:

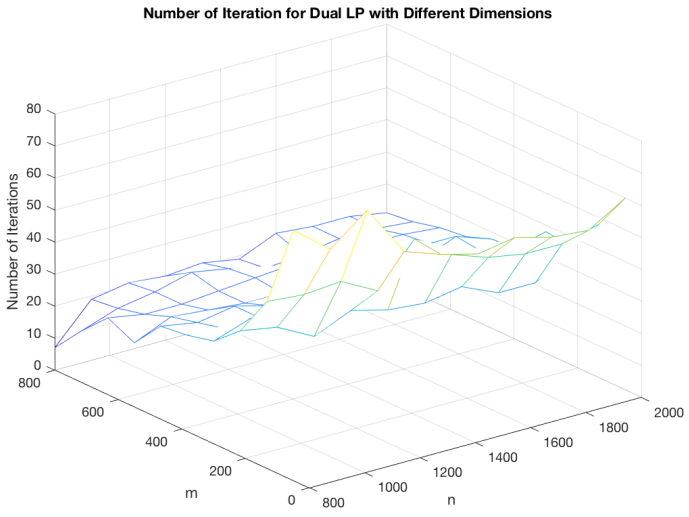

Fig. 13: Convergence of Steps with Different Sizes of A

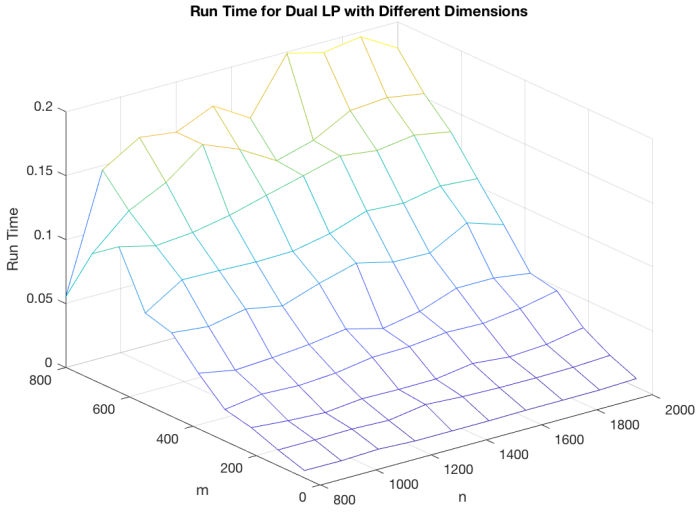

Fig. 14: Convergence of Run Time with Different Sizes of A

Figure 13 shows that Algorithm 2 converges in steps for $50 \leq m \leq 800$ and $800 \leq n \leq 2000$. We observe the following patterns:

- In general, the number of iterations decreases in m, i.e. the number of decision variables in dual form; convergence is unstable when m is small (in our experiment, $m < 100$).
- No clear pattern is observed associated with n, i.e. the number of constraints in dua forml; in particular, the number of iterations increases slightly in n when m is large (in our experiment $m \geq 600$).

Figure 14 shows that Algorithm 2 converges in run time, and we further observe the following patterns:

- Run time increases significantly in the number of decision variables in dual form (m).
- Run time increases slightly in the number of constraints in dual form (n).

The above results are consistent with the intuition that the time to converge increases with the size of A.

2.2.3 Preconditioning

In the case that $m > n$, Algorithm 2 does converge, and even in fewer steps in our simulations, similar to the result of the primal form. However, $A_{m \times n}$

is singular, and thus the result may not be reliable. Therefore we need to precondition A and $\mathbf{b}$ by $(AA^T)^{-1/2}$. Since Algorithm 2 performs stably with large m and n, here we take $A \in \mathbb{R}^{800\times 600}$. Figure 15 and Figure 16 illustrates the convergence results. The preconditioned problem does not converge in 3000 steps. However, the relative residue decreases sharply after the first iteration and then remains below 0.005 afterwards. In applications that require less accuracy, Algorithm 2 with preconditioning the problem may be effective and accurate.

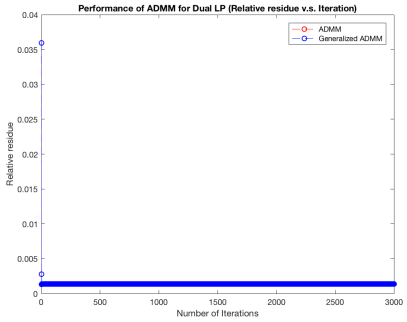

Fig. 15: Convergence of Steps(m=800,n=600)

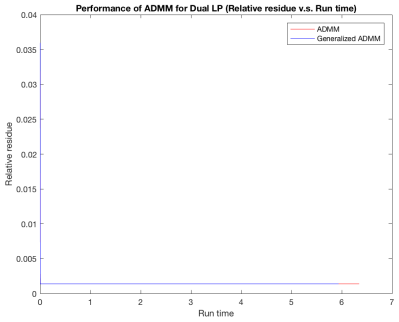

Fig. 16: Convergence of Time(m=800,n=600)

2.3 Interior Point ADMM for Primal

2.3.1 Problem setting

The problem starts with $\mathbf{x}_1^0 = \mathbf{1}$, $\mathbf{x}_2^0 = \mathbf{x}_1^0$, $\mathbf{y}^0 = \mathbf{0}$ and $\mathbf{s}^0 = \mathbf{1}$. The iteration will terminate at step k once the following conditions are satisfied

$$\frac{||A\mathbf{x}_1^k - \mathbf{b}|| + ||\mathbf{x}_1^k - \mathbf{x}_2^k||}{1 + ||\mathbf{b}||} < tol, \tag{3}$$

$$||\mathbf{c}^T\mathbf{x}_1 - \mathbf{b}^T\mathbf{y} - n\mu|| < tol. \tag{4}$$

The sencond condition is derived from the optimality conditions. We still define the relative residue as the sum of norm of residues of the two constraints $A\mathbf{x}_1 - \mathbf{b} = \mathbf{0}$, $\mathbf{x}_1 - \mathbf{x}_2 = \mathbf{0}$, divided by the norm of $\mathbf{b}$.

2.3.2 Sensitivity analysis of A

We first explore the performance of Algorithm 3 with different sizes of A. The convergence sensitivity results are summarized below in Figure 17 and Figure 18. Take $\beta = 1.00$, $\mu = 1.00$.

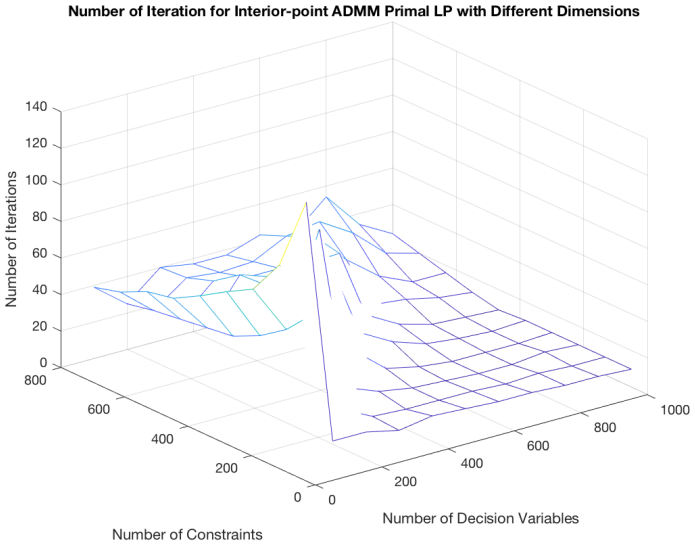

Fig. 17: Convergence of Steps with Different Sizes of A

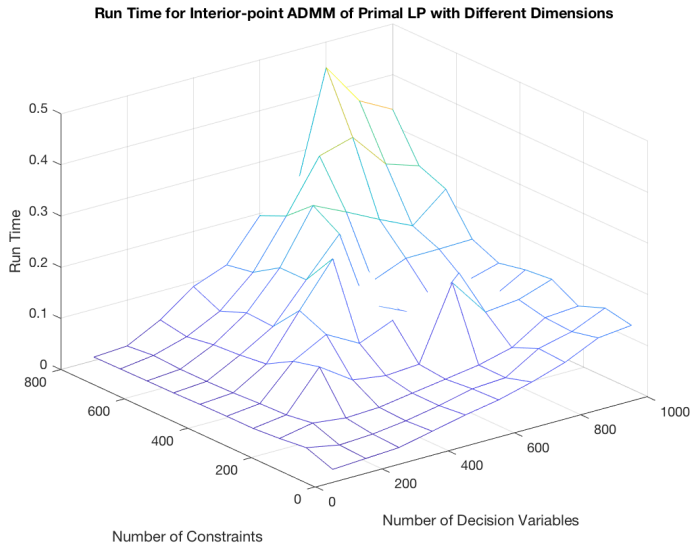

Fig. 18: Convergence of Run Time with Different Sizes of A

From Figure 17, we observe the following patterns

- Algorithm 3 converges quickly when $m \leq n$, and the number of iterations increases in the number of constraints (m).
- When $m > n$, there is no clear pattern associated with m or n; the number of steps required to converge fluctuates with different sizes of A.

Figure 18 shows that run time increases in both m and n, which is consistent with intuition.

2.3.3 Convergence patterns

Next, we compare the convergence performance of Algorithm 3 (Interior Point ADMM for Primal) to that of Algorithm 1 (Basic Primal ADMM), by taking $A \in \mathbb{R}^{60\times 80}$, $A \in \mathbb{R}^{300\times 400}$, $A \in \mathbb{R}^{400\times 300}$. Results are illustrated in Figure 19 to Figure 24.

We observe the following patterns:

- Algorithm 3 performs better than Algorithm 1 with small-sized A ($m \leq n$)(Figure 19, Figure 20), and is more stable in terms of steps and run time when we simulate 10 times.
- Algorithm 3 outperforms Algorithm 1 when $m \leq n$, while Algorithm 1 converges faster when $m > n$.

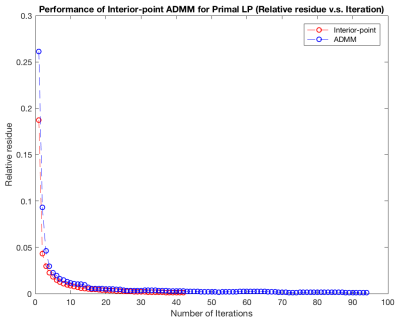

Fig. 19: Convergence of Steps(m=60,n=80)

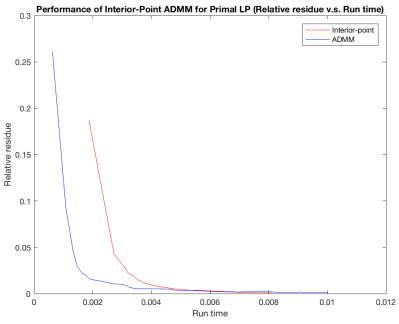

Fig. 20: Convergence of Time(m=60,n=80)

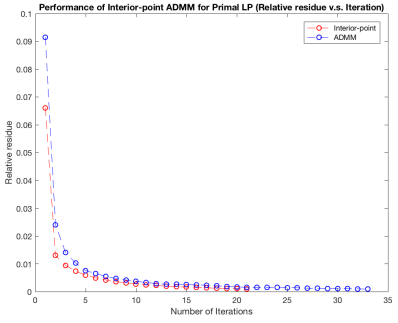

Fig. 21: Convergence of Steps(m=300,n=400)

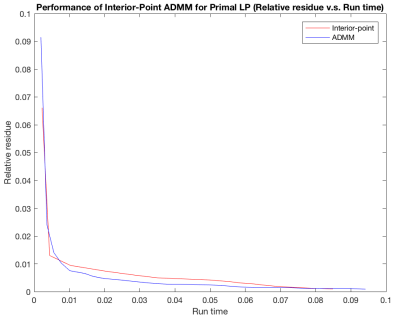

Fig. 22: Convergence of Time(m=300,n=400)

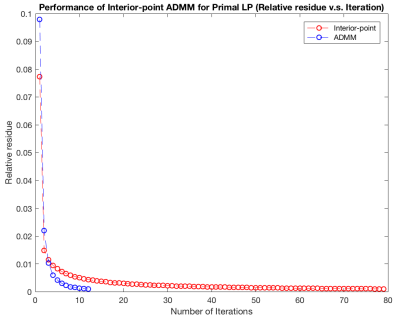

Fig. 23: Convergence of Steps(m=400,n=300)

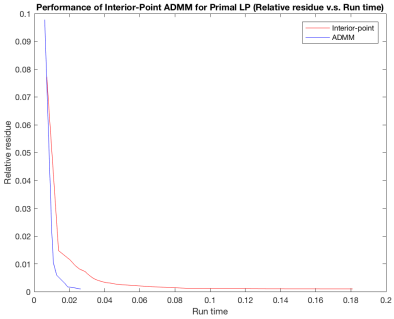

Fig. 24: Convergence of Time(m=400,n=300)

In summary, Algorithm 3 is especially applicable to linear programming problems with fewer constraints and decision variables. But it perform especially inefficient when $m > n$.

2.3.4 Outer-iteration

Finally, we implement the Outer-Iteration process by gradually reducing μ at each step. We start with $\mu^0 = 1.00$, and try different β and γ to do sensitivity analysis. As discussed above, Algorithm 3 performs well when $m \leq n$. WLOG, we take $A \in \mathbb{R}^{600\times 800}$.

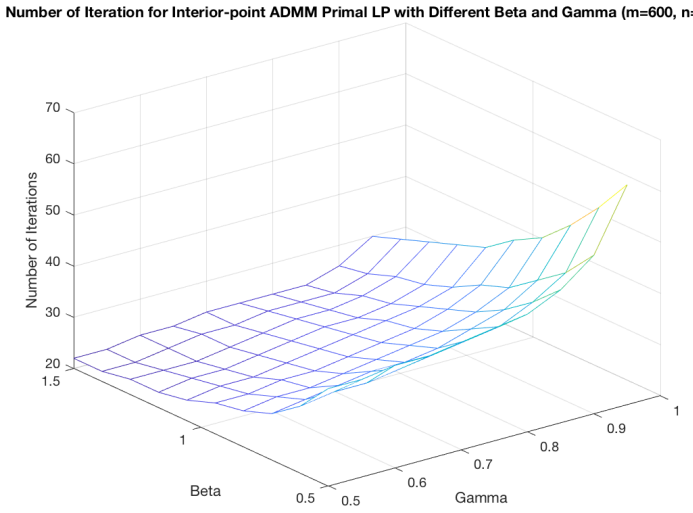

Fig. 25: Convergence of Steps with Different β and γ

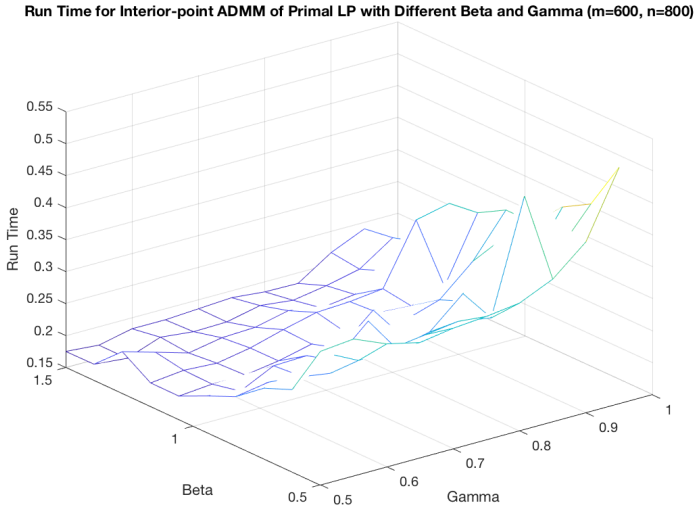

Fig. 26: Convergence of Run Time with Different β and γ

Figure 25 and Figure 26 show that in general, the number of iterations as well as run time decreases in β, and increases in γ. The first observation is intuitive, because when the "penalty parameter" is large, the algorithm would converge faster.

2.4 Interior Point ADMM for Dual

2.4.1 Problem setting

The problem starts with $\mathbf{y}^0 = \mathbf{0}$, $\mathbf{s}^0 = \mathbf{1}$ and $\mathbf{x}^0 = \mathbf{1}$. The iteration will terminate at step k once the following conditions are satisfied

$$\frac{||A^T\mathbf{y}^k + \mathbf{s}^k - \mathbf{c}||}{1 + ||\mathbf{c}||} < tol, \tag{5}$$

$$||\mathbf{c}^T\mathbf{x} - \mathbf{b}^T\mathbf{y} - n\mu|| < tol. \tag{6}$$

2.4.2 Sensitivity analysis of A

We first explore the performance of Algorithm 4 with different sizes of A. To preserve the non-singularity of A, we set $50 \leq m \leq 800$ and $800 \leq n \leq 2000$. The convergence sensitivity results are summarized below in Figure 27 and Figure 28. Take $\beta = 1.00$, $\mu = 1.00$.

From Figure 27, we observe that the number of iterations experiences a sharp increase when the number of decision variables in dual form, m, decreases to a small number (in our experiment below 400). The observation is to some degree consistent with the finding in Algorithm 3.There is no clear pattern associated with the number of constraints n.

Figure 28 shows that in general, run time increases in both n and m; except that for each n, we observe an increase in run time when m is small (< 400).

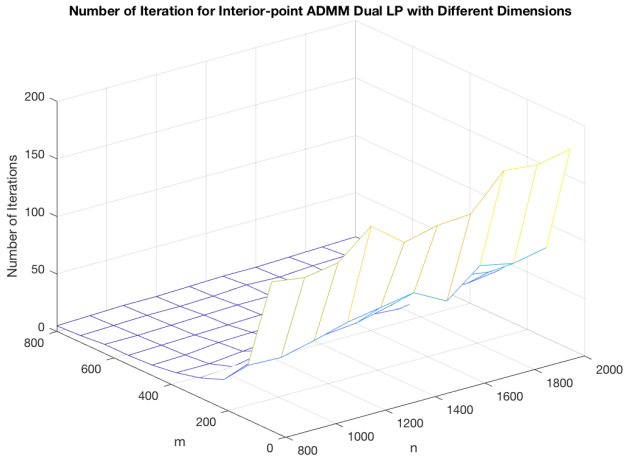

Fig. 27: Convergence of Steps with Different β and γ

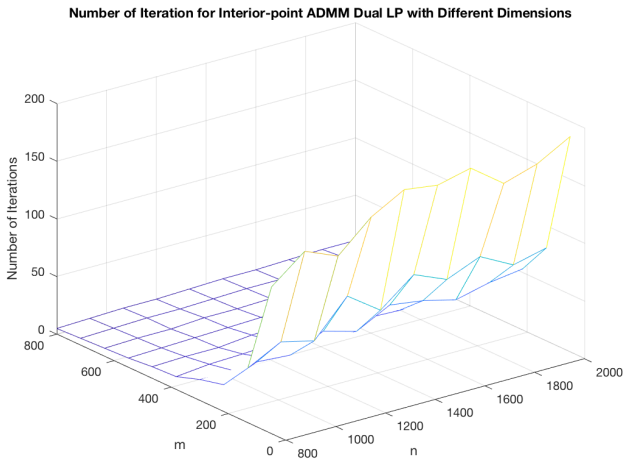

Fig. 28: Convergence of Run Time with Different β and γ

2.4.3 Convergence patterns

Next, we compare the convergence performance of Algorithm 4 (Interior Point ADMM for Dual) to that of Algorithm 2 (Basic Dual ADMM), by taking $A \in \mathbb{R}^{60\times 80}$, and $A \in \mathbb{R}^{600\times 1000}$. We also precondition the problem and implement on $A \in \mathbb{R}^{80\times 60}$. Results are illustrated in Figure 29 to Figure 34. We observe the following patterns:

- In general, Algorithm 4 outperforms Algorithm 2 in terms of steps and run time, especially with small-sized A ($m \leq n$) (Figure 29, Figure 30, Figure 31, Figure 32).
- After implementing preconditioning, neither Algorithm 2 or Algorithm 4 converges; however, Algorithm 4 is more accurate than Algorithm 2 with small relative residues (Figure 33, Figure 34).

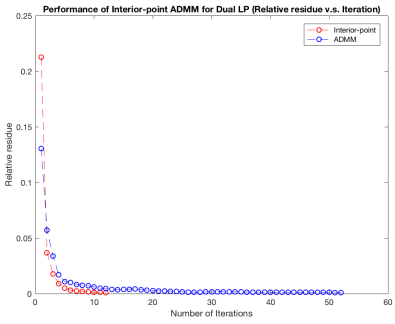

Fig. 29: Convergence of Steps(m=60,n=80)

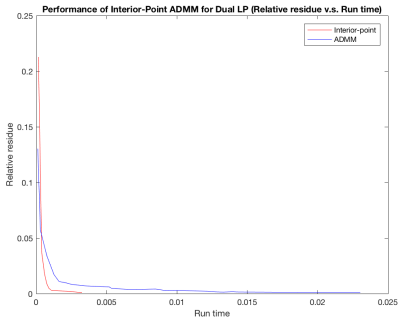

Fig. 30: Convergence of Time(m=60,n=80)

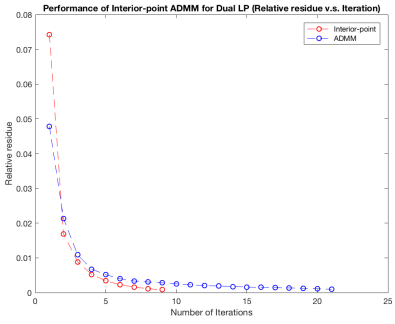

Fig. 31: Convergence of Steps(m=600,n=1000)

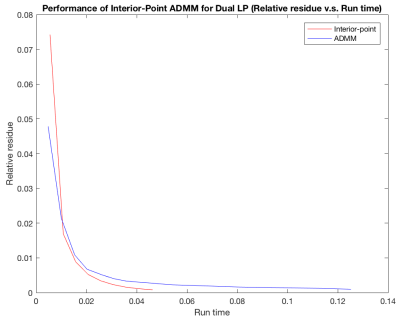

Fig. 32: Convergence of Time(m=600,n=1000)

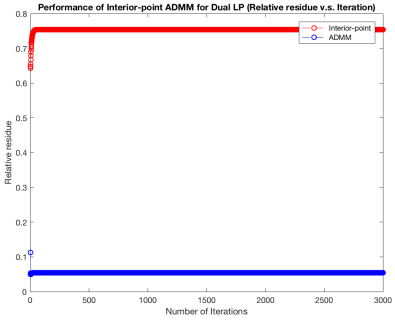

Fig. 33: Convergence of Steps(m=80,n=60)

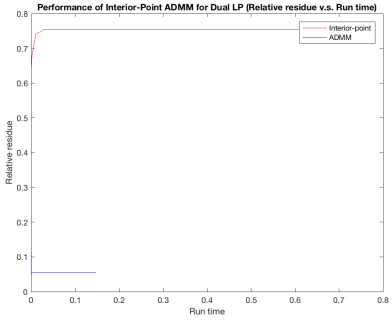

Fig. 34: Convergence of Time(m=80,n=60)

2.4.4 Outer-iteration

Finally, we implement the Outer-Iteration process using a similar approach in Algorithm 3. WLOG, we take $A \in \mathbb{R}^{600\times 1000}$.

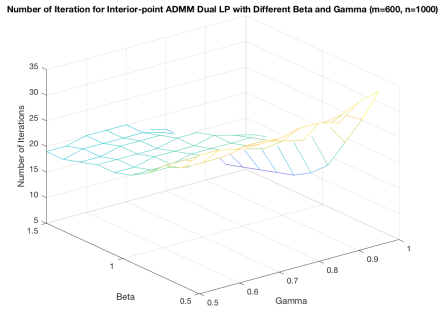

Fig. 35: Convergence of Steps

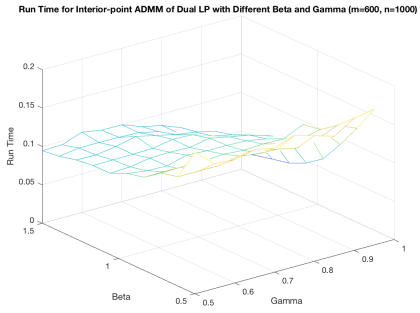

Fig. 36: Convergence of Time

Figure 35 and Figure 36 show that in general, the number of iterations decreases in both β and γ. The first observation is intuitive as discussed in Algorithm 3.

2.5 Multiblock ADMM for Primal

2.5.1 Problem setting

The problem starts with $\mathbf{x}_1^0 = \mathbf{1}$, $\mathbf{x}_2^0 = \mathbf{x}_1^0$, $\mathbf{y}^0 = \mathbf{0}$, and $\mathbf{s}^0 = \mathbf{1}$. Splitting the problem into N blocks, the iteration will terminate at step k once the following condition is satisfied

$$\frac{||\sum_{i=1}^{N} A_i\mathbf{x}_{1,i}^k - \mathbf{b}|| + ||\sum_{i=1}^{N}(\mathbf{x}_{1,i}^k - \mathbf{x}_{2,i}^k)||}{1 + ||\mathbf{b}||} < tol. \tag{7}$$

For the rest of this subsection, we take $\beta = 1.00$.

2.5.2 Sensitivity analysis of the number of blocks

Based on the findings in Algorithm 1, we would like to evaluate the performance of splitting into different number blocks where A_i has full row rank or column rank. Therefore, WLOG, we take $A \in \mathbb{R}^{1000\times 8000}$, and split $\mathbf{x}_1$ and $\mathbf{x}_2$ each into 2, 4, 8, 16, 32 blocks. We siminulate 30 times, and the average convergence results are illustrated Table 1. We also include the result using Algorithm 1 for comparison.

Table 1: Average Convergence Performance by Splitting into 2,4,8,16, 32 Blocks

N blocks	Iteration	Run time	Objective value
RP-ADMM			
2	18	5.31e-01	5.25e+02
4	26	7.84e-01	2.60e+03
8	261	8.20e+00	3.12e+03
16	247	7.52e+00	4.31e+03
32	252	8.93e+00	4.38e+03
Fixed-order			
2	18	4.72e-01	5.17e+02
4	27	7.62e-01	2.65e+03
8	274	7.86e+00	3.08e+03
16	256	7.80e+00	4.28e+03
32	254	8.95e+00	4.41e+03
Algorithm 1			
/	15	7.62e-01	2.65e+03

We observe the following patterns

- In our experiment, both randomly permuted and fixed-order ADMM converge; no clear difference in performance efficiency is observed.
- Iteration step decreases significantly when splitting $\mathbf{x}_1$ and $\mathbf{x}_2$ each into 2 or 4 blocks, further run time decreases by around 1/2 by splitting into 2 blocks; however, there is an sharp increase in both iteration steps and run time by around 10 times when splitting into 8 blocks, and the convergence performance remains stable for 16 and 32 blocks.
- The converging optimal objective value is close to that of Algorithm 1 when splitting into 4 or 8 blocks.

In other words, Algorithm 5, where we either randomly permute the update order of $\{\mathbf{x}_{1,i}\}_{i\leq N}$ and $\{\mathbf{x}_{2,i}\}_{i\leq N}$ or update them sequentially, converges quickly in steps when $\{A_i\}_{i\leq N}$ has full row rank, while it increases computation complexitiy when $\{A_i\}_{i\leq N}$ has full column rank. In particular the algorithm converges one time faster than Algorithm 1 when splitting into 2 blocks.

2.6 Multiblock ADMM for Dual

2.6.1 Problem setting

The problem starts with $\mathbf{y}^0 = \mathbf{0}$, $\mathbf{s}^0 = \mathbf{1}$ and $\mathbf{x}^0 = \mathbf{1}$. For the rest of this subsection, we take $\beta = 1.00$. Splitting the problem into N blocks, the iteration will terminate at step k once the following condition is satisfied

$$\frac{||\sum_{i=1}^{N}(A^T)_i\mathbf{y}_i^k + \mathbf{s}^k - \mathbf{c}||}{1 + ||\mathbf{c}||} < tol. \tag{8}$$

2.6.2 Sensitivity analysis of the number of blocks

One important application of Algorithm 6 is that it could treat problems where $m > n$, because by splitting A^T horizontaly, i.e. "splitting m", we can ensure the non-singularity of $(A_i^T)^T$. Hence we take $A \in \mathbb{R}^{8000\times4000}$, and $A \in \mathbb{R}^{4000\times4000}$, and split $\mathbf{y}$ into 2, 4, 8 blocks for the first case and 2, 4, 8, 16 blocks for the second case. We siminulate 30 times, and the average convergence results are summarized in Table 2 and Table 3.

Table 2: Average Convergence Performance by Splitting into 2,4,8 Blocks (m=8000, n=4000)

N blocks	Iteration	Run time	Objective value	Relative residue
RP-ADMM				
2	4	4.47e+00	1.47e+03	9.17e-04
4	18	7.32e+00	2.18e+03	9.55e-04
8	19	2.86e+00	8.47e+03	9.97e-04
Fixed-order				
2	4	4.39e+00	1.47e+03	9.17e-04
4	20	8.20e+00	1.83e+03	9.09e-04
8	21	3.12e+00	1.65e+03	9.94e-04

Table 3: Average Convergence Performance by Splitting into 2,4,8,16 Blocks (m=4000, n=4000)

N blocks	Iteration	Run time	Objective value	Relative residue
RP-ADMM				
2	42	6.25e+00	2.91e+03	9.58e-04
4	49	3.27e+00	7.84e+03	9.90e-04
8	52	1.92e+00	9.38e+03	9.86e-04
16	63	1.98e+00	8.73e+03	9.75e-04
Fixed-order				
2	42	6.45e+00	2.91e+03	9.58e-04
4	49	3.54e+00	7.58e+03	9.99e-04
8	53	2.00e+00	9.14e+03	9.81e-04
16	57	1.66e+00	9.77e+03	9.80e-04
Algorithm 2				
/	5	2.10e+00	7.66e+02	7.78e-04

Table 2 shows that Algorithm 6 converges by both RP-ADMM and fixed-order ADMM. However, an increase in N does not significantly improve the performance in either run tim and convergence steps.

Table 3 shows that Algorithm 6 is generally outperformed by Algorithm 2 when $m \leq n$. Specifically, more steps are required to converge when splitting into 2,4,8 or 16 blocks; run time increases with 2 and 4 blocks, and decreases slightly with 8 and 16 blocks, compared to Algorithm 2. Finally, in general, Algorithm 6 produces unstable optimal objective value.

References

[1] Chen, He, Ye, and Yuan. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*. 155 (1-2), 2016, 57-79.

[2] Boyd, Parikh, Chu, Peleato and Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*. Volume3, Issue1, January 2011, Pages 1-122.

[3] Giselsson and Boyd. Diagonal Scaling in Douglas-Rachford Splitting and ADMM. *53rd IEEE Conference on Decision and Control*. December 15-17, 2014. Los Angeles, California, USA.

[4] Sun, Luo and Ye. On the expected convergence of randomly permuted ADMM. *arXiv preprint arXiv:1503.06387*, 2015.

[5] O'Donoghue, Chu, Parikh and Boyd. Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding. *Journal of Optimization Theory and Applications*. June 2016, Volume 169, Issue 3, pp 1042C1068.

[6] Eckstein and Yao. Understanding the Convergence of the Alternating Direction Method of Multipliers: Theoretical and Computational Perspectives.
`https://pdfs.semanticscholar.org`