Optimization Methods for Multi-User Detection in Uplink Grant-Free NOMA

Tadi Ravi Teja Reddy | 505227246 | ECE 236C

1 Introduction

Multiple access has become one of the key technologies to distinguish different generations of wireless systems in the history of wireless communications [1]. Frequency division multiple access (FDMA) in 1G, time division multiple access (TDMA) in 2G, code division multiple access (CDMA) in 3G, and orthogonal frequency division multiple access (OFDMA) in 4G are adopted in the evolution history of wireless communications. These conventional multiple access schemes belong to the category of orthogonal multiple access (OMA), which can avoid or reduce inter-user interference by orthogonal resource allocation. However, the number of supportable users is strictly limited by the number of available orthogonal resources in OMA, which makes it difficult to meet the key requirements of massive connectivity for 5G [1]. To address this issue, non-orthogonal multiple access (NOMA) has been actively investigated [1,2], which can realize overloading by introducing some controllable interference at the cost of slightly increased receiver complexity.

As with all the NOMA schemes, multi-user detection (MUD) is a key performance enabler for grant-free NOMA systems [3]. The application of MUD to the Internet-of-Things (IoT) scenario would require an efficient handling of the case where the number of active users is much smaller than the total number of users. This user sparsity enables the application of conventional compressive sensing (CS) algorithms like the orthogonal matching pursuit (OMP) and the iterative support detection (ISD) using truncated basis pursuit (BP) [4] for joint user activity and data detection. These CS algorithms have been utilized by prior works on grant-free NOMA techniques [3], where MUD is realized independently for each transmission time interval. However, in IoT transmission, signals in consecutive transmission time intervals may be correlated, which necessitates novel algorithms leveraging this correlation for better performance. In [5], an alternative direction method of multipliers (ADMM)-based MUD solution in grantfree NOMA systems is proposed. They show that their approach outperforms the state-of-the-art MUD methods since they exploits the knowledge of partial support set and the prior information of the transmitted symbols.

2 Objective

The objective of this report is to investigate the ADMM proposed in [5] and to make some improvements and present a comparative analysis with other optimization methods. In the next section, we define the model of our system and formulate our problem of interest. In Section 4, we present the ADMM algorithm presented in [5] and it's modifications. In Section 5 we try to tackle a relaxed system model with Proximal gradient based methods [6–8]. Finally, in Section 6 we present simulations to evaluate the algorithms discussed in Section 4 and 5.

3 System Model and Problem Formulation

We consider an uplink NOMA system with one Base station (BS) and K users and M active users, where all nodes are equipped with a single antenna. We assume that the system uses N orthogonal subbands, where $N < K$, i.e., the overloaded system. The data symbol is spread and transmitted over N subbands. The received signal at the BS on the n-th subcarrier within time interval t is expressed as

$$y_n(t) = \sum_{k=1}^{K} g_{nk}(t) s_{nk} x_k(t) + v_n(t), \;\; n = 1, 2, \ldots, N \tag{1}$$

where $x_k(t)$ is the transmit symbol for active user k at time t taken from the complex-constellation set $\mathcal{X}$, s_{nk} is the n-th component of spreading sequence s_k of length N, where $x_k(t)$ is modulated onto. $g_{nk}(t)$ is the channel gain of user k on the n-th subcarrier at time t and it is an identically and independent distributed (i.i.d.) complex Gaussian variable with zero mean and unit variance. Finally, $v_n(t)$ is the sample of additive white Gaussian noise with zero mean and variance σ^2 on subcarrier n and at time t. Stacking the received signals over all N subcarriers, the received vector at time t can be written as

$$\tilde{\mathbf{y}}_\mathbf{t} = \tilde{\mathbf{H}}_\mathbf{t} \tilde{\mathbf{x}}_\mathbf{t} + \tilde{\mathbf{v}}_\mathbf{t}, \;\; t = 1, \ldots, T \tag{2}$$

here $\tilde{\mathbf{x}}_\mathbf{t} \in \mathbb{C}^K$, $\tilde{\mathbf{y}}_\mathbf{t} \in \mathbb{C}^N$ are the transmitted and received signal vector respectively at time t and $\tilde{\mathbf{H}}_\mathbf{t} \in \mathbb{C}^{N \times K}$ is an equivalent channel matrix. We reformulate this problem as follows to restrict it to the real domain

$$\mathbf{y_t} = \begin{bmatrix} \mathbb{R}\text{e}\{\tilde{\mathbf{y}}_\mathbf{t}\} \\ \mathbb{I}\text{m}\{\tilde{\mathbf{y}}_\mathbf{t}\} \end{bmatrix}, \;\; \mathbf{x_t} = \begin{bmatrix} \mathbb{R}\text{e}\{\tilde{\mathbf{x}}_\mathbf{t}\} \\ \mathbb{I}\text{m}\{\tilde{\mathbf{x}}_\mathbf{t}\} \end{bmatrix}, \;\; \mathbf{v_t} = \begin{bmatrix} \mathbb{R}\text{e}\{\tilde{\mathbf{v}}_\mathbf{t}\} \\ \mathbb{I}\text{m}\{\tilde{\mathbf{v}}_\mathbf{t}\} \end{bmatrix} \tag{3}$$

$$\mathbf{H_t} = \begin{bmatrix} \mathbb{R}\text{e}\{\tilde{\mathbf{H}}_\mathbf{t}\} & \mathbb{R}\text{e}\{\tilde{\mathbf{H}}_\mathbf{t}\} \\ \mathbb{I}\text{m}\{\tilde{\mathbf{H}}_\mathbf{t}\} & \mathbb{I}\text{m}\{\tilde{\mathbf{H}}_\mathbf{t}\} \end{bmatrix} \tag{4}$$

here, $\mathbf{x_t} \in \mathbb{R}^{2K}$, $\mathbf{y_t} \in \mathbb{R}^{2N}$ and $\mathbf{H_t} \in \mathbb{R}^{2N \times 2K}$. We rewrite (2) as follows

$$\mathbf{y_t} = \mathbf{H_t} \mathbf{x_t} + \mathbf{v_t}, \;\; t = 1, \ldots, T \tag{5}$$

As discussed in Section 1, MUD applications to the Internet of things (IoT) scenario would require a sparse reconstruction of $\mathbf{x}_t$ from $\mathbf{y}_t$ by solving the following basis pursuit denoising (BPDN) problem

$$\min_{\mathbf{x}} \;\; \lambda_t ||\mathbf{x}||_1 + 0.5 ||\mathbf{y_t} - \mathbf{H_t}\mathbf{x}||_2^2 \tag{6}$$

Here, λ_t is a regularization parameter. Since we desire for a solution that achieves exact reconstruction using fewer measurements, we look into the modified-CS idea by exploiting the partial known support set τ, which denotes the set of location of nonzero values in the signal $\mathbf{x_t}$. To that end, the sparse recovery problem becomes the one that tries to find the vector $\mathbf{x_t}$ that is sparsest outside the set τ, that is

$$\min_{\mathbf{x}} \;\; \lambda_t ||\mathbf{x}||_{1,\mathbf{w}} + \frac{1}{2} ||\mathbf{y_t} - \mathbf{H_t}\mathbf{x}||_2^2, \tag{7}$$

where $||\mathbf{x}||_{1,\mathbf{w}} = \sum_{i=1}^{K} w_i |x_i|$ with $w_i = 0, \;\; i \in \tau, \;\; w_i = 1, \;\; i \notin \tau$. Note that the first term in (7) does not penalize the nonzero terms whose locations are known. However, the

problem (7) puts no cost on $\mathbf{x}$ except the cost imposed by the data term. Thus, when very few measurements are available or when the noise is large, $\mathbf{x}$ can become larger than required. To address this, when reliable prior signal value knowledge is available, [9] exploit this knowledge to design improved solutions and solve the "regularized modified-BPDN" problem formulated as:

$$\min_{\mathbf{x}} \ \lambda_t ||\mathbf{x}||_{1,\mathbf{w}} + \frac{1}{2}||\mathbf{y_t} - \mathbf{H_t}\mathbf{x}||_2^2 + \frac{\alpha_t}{2}||\mathbf{x} - \hat{\mathbf{x}}_\mathbf{t}||, \tag{8}$$

where $\hat{\mathbf{x}}_\mathbf{t}$ denotes an erroneous estimate (prior information) of the signal values $\mathbf{x}_t$ obtained in the previous iteration/time-slot and is used to achieve better reconstruction quality.

4 ADMM

We adopt the regularized modified BPDN algorithm as our CS algorithm to solve the MUD problem in uplink grant-free systems, which exploits both the partial support and the transmitted signal estimate as prior information. To be able to apply the ADMM algorithm, the problem (8) can be equivalently written as

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{z}} \ \frac{1}{2}||\mathbf{y_t} - \mathbf{H_t}\mathbf{x}||_2^2 + \frac{\alpha_t}{2}||\mathbf{x} - \hat{\mathbf{x}}_\mathbf{t}|| + \lambda_t||\mathbf{z}||_{1,\mathbf{w}} \\ &\text{s.t } \mathbf{x} = \mathbf{z} \end{aligned} \tag{9}$$

We introduce a dual variable $\mathbf{t}_t$ and a penalty parameter ρ_t and we express the augmented Lagrangian is as

$$\mathcal{L}(\mathbf{x}, \mathbf{z}, \mathbf{t}_t) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{t}_t^T(\mathbf{x} - \mathbf{z}) + \frac{\rho_t}{2}||\mathbf{x} - \mathbf{z}||_2^2 \tag{10}$$

here, $f(\mathbf{x}) = \frac{1}{2}||\mathbf{y_t} - \mathbf{H_t}\mathbf{x}||_2^2 + \frac{\alpha_t}{2}||\mathbf{x} - \hat{\mathbf{x}}_\mathbf{t}||$ and $g(\mathbf{z}) = \lambda_t||\mathbf{z}||_{1,\mathbf{w}}$. The ADMM algorithm decomposes the main optimization problem in (9) into easier and smaller local subproblems such that the optimization variables x and z can be computed separately in an alternating fashion. The scaled ADMM form of the problem (9) consists of three iterations:

$$\mathbf{x}_t^{[k+1]} = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{\rho_t}{2}||\mathbf{x} - \mathbf{z}_t^{[k]} + \mathbf{u}_t^{[k]}||_2^2 \right), \tag{11}$$

$$\mathbf{z}_t^{[k+1]} = \arg\min_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\rho_t}{2}||\mathbf{x}_t^{[k+1]} - \mathbf{z} + \mathbf{u}_t^{[k]}||_2^2 \right), \tag{12}$$

$$\mathbf{u}_t^{[k+1]} = \mathbf{u}_t^{[k]} + \mathbf{x}_t^{[k+1]} - \mathbf{z}_t^{[k+1]}, \tag{13}$$

where $\mathbf{u}_t = \frac{\mathbf{t}_t}{\rho_t}$ is the scaled dual variable. The optimal $\mathbf{x}_t$ can be computed from (11) and it is given as

$$\mathbf{x}_t^{[k+1]} = (\mathbf{H}_t^T\mathbf{H}_t + (\rho_t + \alpha_t)\mathbf{I}_k)^{-1} \times \left(\mathbf{H}_t^T\mathbf{y}_t + \alpha_t(\hat{\mathbf{x}}_\mathbf{t}) + \rho_t(\mathbf{z}_t^{[k]} - \mathbf{u}_t^{[k]}) \right) \tag{14}$$

Note: The update in (14) requires us to calculate the inverse of $\mathbf{H}_t^T\mathbf{H}_t + (\rho_t + \alpha_t)\mathbf{I}_k$ in every iteration, which could be quite expensive. We reduce the computational cost, by using eigenvalue decomposition beforehand and only calculating the inverse of a diagonal matrix in every iteration.

Even though the problem (12) is not differentiable, the closed form solution of (12) can be computed through sub-differential calculus, i.e., using the soft thresholding operator, and the optimal $\mathbf{z}_t^{[k+1]}$ is expressed as:

$$z_i^{[k+1]}(t) = \max\left\{x_i^{[k+1]}(t) + u_i^{[k]}(t) - \frac{\lambda_t w_i}{\rho_t}, 0\right\} - \max\left\{-x_i^{[k+1]}(t) - u_i^{[k]}(t) - \frac{\lambda_t w_i}{\rho_t}, 0\right\} \tag{15}$$

Algorithm 1 ADMM Algorithm

1: Initialize $\tau^{[0]} = \emptyset$, $k = 1$, $\alpha_t^{[0]} = 0$, $\hat{\mathbf{x}}_\mathbf{t} = \mathbf{0}$, $\mathbf{r}_t^{[0]} = \infty$, $\mathbf{s}_t^{[0]} = \infty$, $\mathbf{z}_\mathbf{t}^{[0]} = \mathbf{0}$, $\mathbf{u}_\mathbf{t}^{[0]} = \mathbf{0}$ and $\mathbf{w}^{[0]}$ is a unit vector.
2: **repeat**
3: **for** $t = 1 : T$ **do**
4: compute $\mathbf{x}_t^{[k]}$, $\mathbf{z}_t^{[k]}$ and $\mathbf{u}_t^{[k]}$ from (14), (15) and (13)
5: **end for**
6: $\mathbf{z}_t^{[k]} = \sum_{t=1}^{T} |\mathbf{z}_t^{[k]}|$.
7: Update the support threshold $\beta^{[k]}$ using the "first significant jump" [4]. In particular, first sort $|\mathbf{z}^{[k]}|$ in the increasing order.($|\mathbf{z}_i^{[k]}|$ denotes the i-th largest component of $\mathbf{z}^{[k]}$ by magnitude). The first significant jump is the smallest i such that

$$|z_{i+1}^{[k]}| - |z_i^{[k]}| > \frac{7}{2N} \frac{||\mathbf{z}^{[k]}||_\infty}{\min(k, 6)} \tag{16}$$

Then, we have $\beta^{[k]} = |z_i^{[k]}|$.
8: Update the support set $\tau^{[k]} = \{i : z_i^{[k]} > \beta^{[k]}\}$, and the signal value $\hat{\mathbf{x}}_\mathbf{t} = \mathbf{z}_t^{[k]}$. Set the weights $w_i^{[k]} = 0$, $i \in \tau^{[k]}$, $w_i^{[k]} = 1$, $i \notin \tau^{[k]}$.
9: compute the primal residual $\mathbf{r}_t^{[k]} = \mathbf{x}_t^{[k]} - \mathbf{z}_t^{[k]}$, and the dual residual $\mathbf{s}_t^{[k]} = -\rho_t(\mathbf{z}_t^{[k]} - \mathbf{z}_t^{[k-1]})$.
10: Update the regularization parameter according to:

$$\alpha_t^{[k]} = (KC_\alpha)/(N(||\mathbf{s}_t^{[k]}||_2 + ||\mathbf{r}_t^{[k]}||_2)). \tag{17}$$

11: $k := k + 1$
12: **until** convergence or maximum number of iterations is reached

The proposed ADMM for our model is given in Algorithm 1. The condition for convergence of the proposed ADMM algorithm is that primal and dual residual norms tend to a small value, i.e., $||\mathbf{r}^{[k]}||_F \leq \epsilon^{primal}$ and $||\mathbf{s}^{[k]}||_F \leq \epsilon^{dual}$ [10], here $\mathbf{r}^{[k]} = [\mathbf{r}_1^{[k]}, \ldots, \mathbf{r}_T^{[k]}]$ and $\mathbf{s}^{[k]} = [\mathbf{s}_1^{[k]}, \ldots, \mathbf{s}_T^{[k]}]$.

Note: Regularization parameter α_t and support set τ vary with every iteration as shown in Algorithm 1. Hence, the objective (8) changes with every iteration.

4.1 Modified-ADMM

A standard extension of ADMM is to use possibly different penalty parameters ρ_t for each iteration, with the goal of improving the convergence. We use the scheme suggested in [10]

$$\rho_t^{k+1} := \begin{cases} \tau^{incr}\rho_t^k, & \text{if } ||r_t^k||_2 \geq \mu||s_t^k||_2 \\ \frac{\rho_t^k}{\tau^{decr}}, & \text{if } ||s_t^k||_2 \geq \mu||r_t^k||_2 \\ \rho_t^k & \text{otherwise} \end{cases} \tag{18}$$

here, $\mu > 1$, $\tau^{incr} > 1$ and $\tau^{decr} > 1$ are parameters. The update in penalty term given in (18) ensures that the primal and dual residual norms are within a factor of μ of one another as they both converge to zero. We define "modified-ADMM" for our model by using penalty parameters updates as given in (18) after step-9 in Algorithm 1 .

4.2 Fast-ADMM

Next, we look at accelerated variant of ADMM proposed in [11], the proposed acceleration is of the form first proposed by Nesterov for gradient descent methods. Since $g(\mathbf{z})$ in our problem is not strongly convex, we need to enforce stability by using a restart rule as described in Algorithm 2. In "fast-ADMM", we introduce the following changes to the updates in (11), (12) and (13).

$$\mathbf{x}_t^{[k+1]} = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{\rho_t}{2}||\mathbf{x} - \hat{\mathbf{z}}_t^{[k]} + \hat{\mathbf{u}}_t^{[k]}||_2^2 \right), \tag{19}$$

$$\mathbf{z}_t^{[k+1]} = \arg\min_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\rho_t}{2}||\mathbf{x}_t^{[k+1]} - \mathbf{z} + \hat{\mathbf{u}}_t^{[k]}||_2^2 \right), \tag{20}$$

$$\mathbf{u}_t^{[k+1]} = \hat{\mathbf{u}}_t^{[k]} + \mathbf{x}_t^{[k+1]} - \mathbf{z}_t^{[k+1]} \tag{21}$$

$\hat{\mathbf{z}}_t^{[k]}$ and $\hat{\mathbf{u}}_t^{[k]}$ are introduced in the fast-ADMM algorithm 2.

Algorithm 2 Fast ADMM Algorithm

Intialize $\tau^{[0]} = \emptyset$, $k = 1$, $\alpha_t^{[0]} = 0$, $\hat{\mathbf{x}}_\mathbf{t} = \mathbf{0}$, $\mathbf{r}_t^{[0]} = \infty$, $\mathbf{s}_t^{[0]} = \infty$, $\mathbf{z}_\mathbf{t}^{[0]} = \mathbf{0}$, $\mathbf{u}_\mathbf{t}^{[0]} = \mathbf{0}$, $\mathbf{w}^{[0]}$ is a unit vector, $\eta \in (0, 1)$, $\hat{\mathbf{z}}_\mathbf{t}^{[0]} = \mathbf{0}$, $\hat{\mathbf{u}}_\mathbf{t}^{[0]} = \mathbf{0}$ and $a_t^{[0]} = 1$
repeat
  for $t = 1 : T$ **do**
    compute $\mathbf{x}_t^{[k]}$, $\mathbf{z}_t^{[k]}$ and $\mathbf{u}_t^{[k]}$ from (19), (20) and (21)
    $c_t^{[k+1]} = \rho_t||\mathbf{u}_t^{[k+1]} - \mathbf{u}_t^{[k]}||^2 + \rho_t||\mathbf{z}_t^{[k+1]} - \mathbf{z}_t^{[k]}||^2$
    if $c_t^{[k+1]} < \eta c_t^{[k]}$ **then**
      $a_t^{[k+1]} = \frac{1+\sqrt{1+4(a_t^{[k]})^2}}{2}$
      $\hat{\mathbf{z}}_\mathbf{t}^{[\mathbf{K+1}]} = \mathbf{z}_\mathbf{t}^{[\mathbf{K+1}]} + \frac{a_t^{[k-1]}}{a_t^{[k+1]}}(\mathbf{z}_\mathbf{t}^{[\mathbf{K+1}]} - \mathbf{z}_\mathbf{t}^{[\mathbf{K}]})$
      $\hat{\mathbf{u}}_\mathbf{t}^{[\mathbf{K+1}]} = \mathbf{u}_\mathbf{t}^{[\mathbf{K+1}]} + \frac{a_t^{[k-1]}}{a_t^{[k+1]}}(\mathbf{u}_\mathbf{t}^{[\mathbf{K+1}]} - \mathbf{u}_\mathbf{t}^{[\mathbf{K}]})$
    else
      $a_t^{[k+1]} = 1$, $\hat{\mathbf{z}}_\mathbf{t}^{[\mathbf{K+1}]} = \mathbf{z}_\mathbf{t}^{[\mathbf{K-1}]}$ and $\hat{\mathbf{y}}_\mathbf{t}^{[\mathbf{K+1}]} = \mathbf{y}_\mathbf{t}^{[\mathbf{K-1}]}$
      $c_t^{[k+1]} = \eta^{-1} c_t^{[k]}$
    end if
  end for
  Follow steps 6 to 10 in Algorithm 1
  $k := k + 1$
until convergence or maximum number of iterations is reached

The restarts in Algorithm 2 is enforced through steps 6 through 13. The choice of η determines the number of restarts for a particular instance, it's suggested to set it to a high value to have fewer restarts [11].

5 Relaxed System Model

In this section, we apply proximal gradient based methods like, Proximal gradient method, Nesterov and FISTA [6–8] to (8). The motivation behind analyzing these methods is the low computationally cost per iteration compared to ADMM based methods for problem (8). Since these methods do not have a splitting variable, we cannot update α_t in every iteration. Hence we relax our model by setting α_t in (8) to a constant value.

Note: Even though α_t is fixed, objective (8) varies with ever iteration since we update the support set τ for every iteration.

5.1 ADMM Based Methods

"modified-ADMM" and "fast-ADMM" will be solved as defined in Section 4, but we won't update α_t in each iteration.

5.2 Proximal Gradient Method

We apply proximal gradient method to our relaxed model, we express our problem as

$$\min_{x} \; f(\mathbf{x}) + g(\mathbf{x}) \tag{22}$$

here, $f(\mathbf{x}) = \frac{1}{2}||\mathbf{y_t} - \mathbf{H_t x}||_2^2 + \frac{\alpha_t}{2}||\mathbf{x} - \hat{\mathbf{x}}_\mathbf{t}||$ and $g(\mathbf{x}) = \lambda_t||\mathbf{x}||_{1,\mathbf{w}}$. We notice that $f(\mathbf{x})$ is convex and differentiable on it's domain and $g(\mathbf{x})$ is convex and has an inexpensive prox-operator (soft thresholding). The proximal gradient decomposes the problem into a single simple step.

$$\mathbf{x}_t^{[k+1]} = \text{prox}_{t_k g}(\mathbf{x}_t^{[k]} - t_k \nabla f(\mathbf{x}_\mathbf{t}^{[\mathbf{k}]})) \tag{23}$$

here, $\text{prox}_{t_k g}(\mathbf{x}_t^{[k]})_i = \max\left\{x_i^{[k+1]}(t) - \lambda_t w_i, 0\right\} - \max\left\{-x_i^{[k+1]}(t) - \lambda_t w_i, 0\right\}$. The proximal gradient for the relaxed system model is described in Algorithm 3

5.3 Nesterov

Nestrov introduces acceleration to proximal gradient method, we replace (23) with

$$\mathbf{y}_t = \mathbf{x}_t^{[k]} + \frac{\gamma^{[k]}\theta^{[k]}}{\gamma^{[k]} + m\theta^{[k]}}(\mathbf{q}_t^{[k]} - \mathbf{x}_t^{[k]}) \tag{25}$$

$$\mathbf{x}_t^{[k+1]} = \text{prox}_{t_k g}(\mathbf{y}_t - t_k \nabla f(\mathbf{y_t})) \tag{26}$$

$$\mathbf{q}_t^{[k+1]} = \hat{\mathbf{x}}_t^{[k]} + \frac{1}{\theta^{[k]}}(\mathbf{x}_t^{[k+1]} - \mathbf{x}_t^{[k]}) \tag{27}$$

for $k \geq 1$, we define $\theta^{[k]}$ as positive root of the quadratic equation

$$\frac{(\theta^{[k]})^2}{t_k} = (1 - \theta^{[k]})\gamma^{[k]} + m\theta^{[k]}, \quad \text{where} \quad \gamma^{[k]} = \frac{(\theta^{[k-1]})^2}{t_{k-1}} \tag{28}$$

Algorithm 3 Proximal Gradient Algorithm

1: Initialize $\tau^{[0]} = \emptyset$, $k = 1$, $\hat{\mathbf{x}}_\mathbf{t} = \mathbf{0}$, $\mathbf{r}_t^{[0]} = \infty$, and $\mathbf{w}^{[\mathbf{0}]}$ is a unit vector.
2: **repeat**
3: **for** $t = 1 : T$ **do**
4: compute $\mathbf{x}_t^{[k]}$ using (23)
5: **end for**
6: $\mathbf{x}_t^{[k]} = \sum_{t=1}^{T} |\mathbf{x}_t^{[k]}|$.
7: Update the support threshold $\beta^{[k]}$ using the "first significant jump" [5]. In particular, first sort $|\mathbf{x}^{[k]}|$ in the increasing order.($|\mathbf{x}_i^{[k]}|$ denotes the i-th largest component of $\mathbf{x}^{[k]}$ by magnitude). The first significant jump is the smallest i such that

$$|x_{i+1}^{[k]}| - |x_i^{[k]}| > \frac{7}{2 * N} \frac{||\mathbf{x}^{[k]}||_\infty}{\min(k, 6)} \tag{24}$$

Then, we have $\beta^{[k]} = |x_i^{[k]}|$.
8: Update the support set $\tau^{[k]} = \{i : x_i^{[k]} > \beta^{[k]}\}$, and the signal value $\hat{\mathbf{x}}_\mathbf{t} = \mathbf{x}_t^{[k-1]}$. Set the weights $w_i^{[k]} = 0$, $i \in \tau^{[k]}$, $w_i^{[k]} = 1$, $i \notin \tau^{[k]}$.
9: compute residual $\mathbf{r}_t^{[k]} = \mathbf{x}_t^{[k]} - \mathbf{x}_t^{[k-1]}$
10: $k := k + 1$
11: **until** convergence or maximum number of iterations is reached

we choose $\mathbf{q}_t^{[0]} = \mathbf{x}_t^{[0]}$ and $\theta^{[0]} \in (0, 1]$. There exists a $m \geq 0$ and $L > 0$ such that $f(\mathbf{x}) - \frac{m}{2}\mathbf{x}^T\mathbf{x}$ and $\frac{m}{2}\mathbf{x}^T\mathbf{x} - g(\mathbf{x})$ are convex. Since $m > 0$, i.e., it is strictly convex in our case, we choose a constant step size $t_k = 1/L$ and set $\theta_0 = \sqrt{m/L}$, then the above set of equations simplify to the following two equations

$$\mathbf{y}_t = \mathbf{x}_t^{[k]} + \frac{1 - \sqrt{m/L}}{1 + \sqrt{m/L}}(\mathbf{x}_t^{[k]} - \mathbf{x}_t^{[k-1]}) \tag{29}$$

$$\mathbf{x}_t^{[k+1]} = \text{prox}_{t_k g}(\mathbf{y}_t - \frac{1}{L}\nabla f(\mathbf{y_t})) \tag{30}$$

We implement "Nesterov" method for our model replacing step-4 of Algorithm 3 with (29) and (30).

5.4 FISTA

"FISTA" is obtained from Subsection 5.3 by using $m = 0$ and we obtain the following two steps

$$\mathbf{y}_t = \mathbf{x}_t^{[k]} + \theta_k(\frac{1}{\theta_{k-1}} - 1)(\mathbf{x}_t^{[k]} - \mathbf{x}_t^{[k-1]}) \tag{31}$$

$$\mathbf{x}_t^{[k+1]} = \text{prox}_{t_k g}(\mathbf{y}_t - t_k\nabla f(\mathbf{y_t})) \tag{32}$$

We incorporate FISTA to our model by replacing step-4 of Algorithm 3 with (31) and (32).
Note: For "Proximal gradient", "Nesterov" and "FISTA" the condition for convergence is that primal residual tends to a small value, i.e., $||\mathbf{r}^{[k]}||_F \leq \epsilon^{primal}$.

6 Simulation Results

In this section, we examine the MUD performance of our algorithms for the original and relaxed system model. Parameters common for both the models are mentioned in Table 1. We generate Spreading sequences based on the pseudo random noise sequence and we will use quaternary phase shift keying (QPSK) as our modulation scheme.

Parameter	Value
K (Number of users)	108
N (Number of orthogonal subbands)	72
M (Number of active users)	20
T (Number of time intervals)	10
Maximum number of iterations	400

Table 1: Parameters for both the models

6.1 Original System Model

In this subsection, we analyze the performance of "ADMM", "modified-ADMM" and "fast-ADMM". The parameters for our system model are mentioned in Table 2. The parameters were chosen for the most optimal performance of each algorithm.

Parameter	Value
C_α	0.1
λ	0.1
μ	10
τ_{incr}, τ_{decr}	2
η	0.9
ϵ^{primal}, ϵ^{dual}	0.001
ρ_t for "ADMM"	1

Table 2: Parameters for original system model

First, we try to compare the performance of "ADMM", "modified-ADMM" and "fast-ADMM" for different values of SNR, we do this by analyzing the variation in BER, number of iterations for convergence and run-time values with SNR for different algorithms. **Note:** we perform 100 random iterations for all the algorithms together to obtain Figure 1, Figure 2 and Table 3, i.e., all three algorithms are evaluated on the same set of instances.

Figure 1 compares BER for different values of SNR, as expected BER decreases with increasing SNR (decreasing noise). We also observe that BER values far all the methods are extremely close for most SNR values. Figure 2 gives us an overview of the number of iterations for convergence for different SNR values. We notice, "modified-ADMM" performs the best for any given SNR and performance of fast-ADMM deteriorates as SNR value increases. This behaviour in "fast-ADMM" could be because of higher number of restarts with larger SNR values. It's also interesting to note that for SNR from 8 to 10, the number of iterations for "ADMM" and "modified ADMM" is the exact same, this is possible if the Lagrange parameter ρ_t does not scale with iterations.

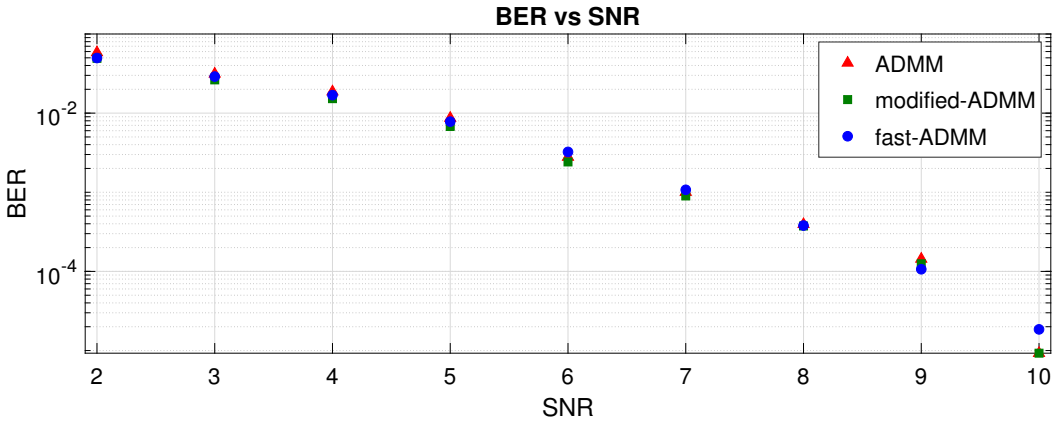

Figure 1: BER vs SNR

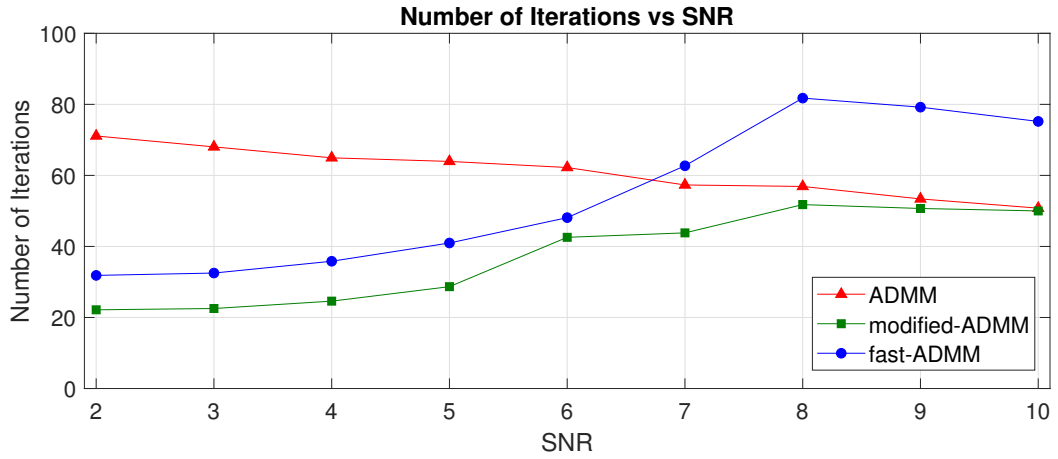

Figure 2: Number of iterations for convergence vs SNR

Next, we look at the average run-time of each of these algorithms with SNR in Table 3. We notice that the run times tally with the observations in Figure 2, "modified-ADMM" performs the best, "fast-ADMM" performs better than "ADMM" for smaller values of SNR but it's run-time got worse for SNR values greater than 6.

SNR	**ADMM**	**modified-ADMM**	**fast-ADMM**
2	1.8	0.6	0.8
3	1.8	0.6	0.9
4	1.8	0.7	1.0
5	1.7	0.8	1.0
6	1.7	1.2	1.3
7	1.6	1.2	1.7
8	1.5	1.4	2.1
9	1.4	1.3	2.0
10	1.4	1.4	2.1

Table 3: Average CPU time in seconds

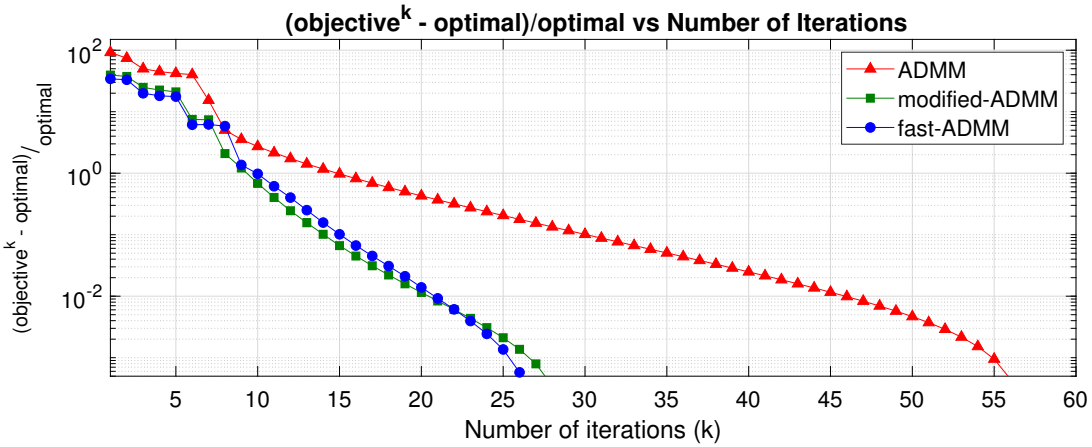

Figure 3: $\frac{objective\ value^{[k]} - optimal\ value}{optimal\ value}$ vs Number of iterations (k)

Next, we look at relative difference in objective and optimal value with number of iterations (k) in Figure 3 for SNR = 5. As expected from the first three simulations,"fast-ADMM" and "modified ADMM" converge faster than the "ADMM".

Note: In Figure 3 we compare the relative difference in objective value from the optimal value, i.e., $\frac{objective\ value^{[k]} - optimal\ value}{optimal\ value}$. As stated in the Sections 4, the objective of our model varies with every iteration, hence we cannot compute the optimal value, hence we set optimal value for each algorithm as the the value the objective takes when the algorithm converges.

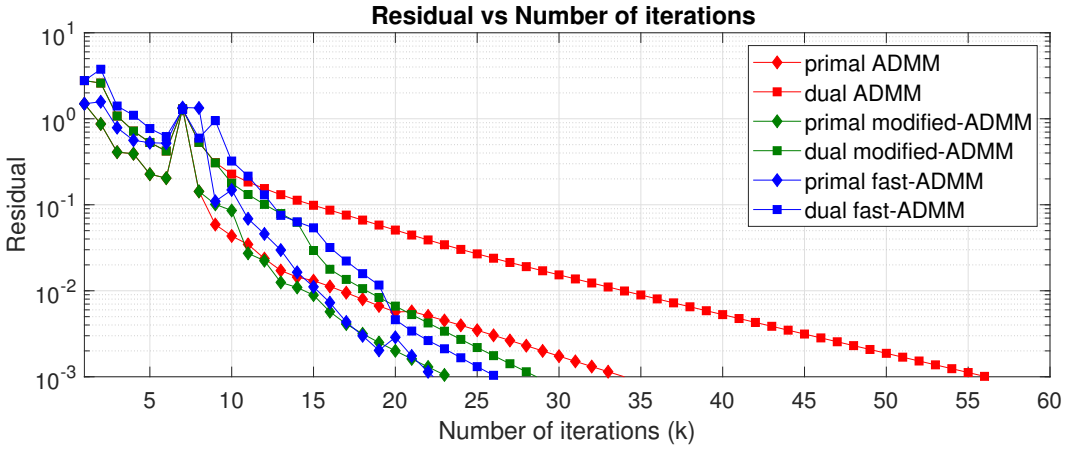

Figure 4: Residuals vs Number of iterations (k)

We also look into the behaviour of primal and dual residuals with number of iterations in Figure 4. We observe that there is a huge variation in the primal and dual residuals of "ADMM", this is significantly reduced in the case of "modified-ADMM" due to the scaling of penalty parameters ρ_t. The residuals of "fast-ADMM" have abrupt changes in the residuals this might be related to the restarts in the Algorithm 2. Lastly, we note that when primal and dual residuals of a particular algorithm drops to a value very small value, the difference in objective and optimal value goes to zero.

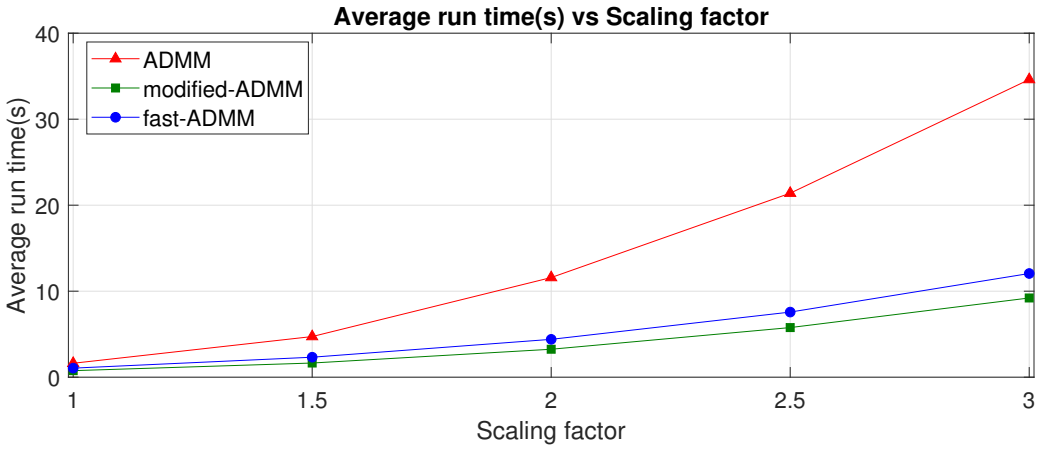

Figure 5: Average CPU run-time vs Scaling factor

For the last part of this subsection, we look into the effect of scaling (we multiply parameters K, N and M with a scaling factor) on the performance of all three algorithms. We set SNR $= 5$ and preform 100 random iterations to average the run-time.

Scaling factor	**ADMM**	**modified-ADMM**	**fast-ADMM**
1.0	1.7	0.8	1.1
1.5	4.7	1.6	2.3
2.0	11.6	3.2	4.4
2.5	21.4	5.8	7.6
3.0	34.6	9.2	12.1

Table 4: Average CPU time in seconds for SNR $= 5$

Table 4 and Figure 5 give the run-time of all three algorithms with the scaling factor. Given the trend, "ADMM" scales the worst of the three algorithms, since the cost per iteration of all three methods should be roughly the same, the difference in performance should be because of the number of iterations required to converge. **Note:** We limit the scaling factor to 3, since it's impractical to have more than 60 active user's at a single base station.

6.2 Relaxed Model

In this Subsection, we analyze the performance of "modified-ADMM", "fast-ADMM", "proximal gradient", "Nesterov" and "FISTA" with each other. The parameters for the relaxed system model is given in Table 5. For "proximal gradient", "Nestrov" and "FISTA" we use a constant step size of $1/L$, here L depends on $\mathbf{H}_t$ and α_t (explained in Subsection 5.3), which are both constant for a particular instance.

Parameter	Value
C_α	0.1
α_t	10
λ	0.1
μ	10
τ_{incr}, τ_{decr}	2
η	0.9
ϵ^{primal}, ϵ^{dual}	0.01
θ_0	0.1
m	0.001

Table 5: Parameters for original system model

In figure 6, we plot BER values for different SNR's, we notice that we get the usual trend of decrease in BER values with increasing SNR, also for a given SNR, the BER of all the algorithms are very close.

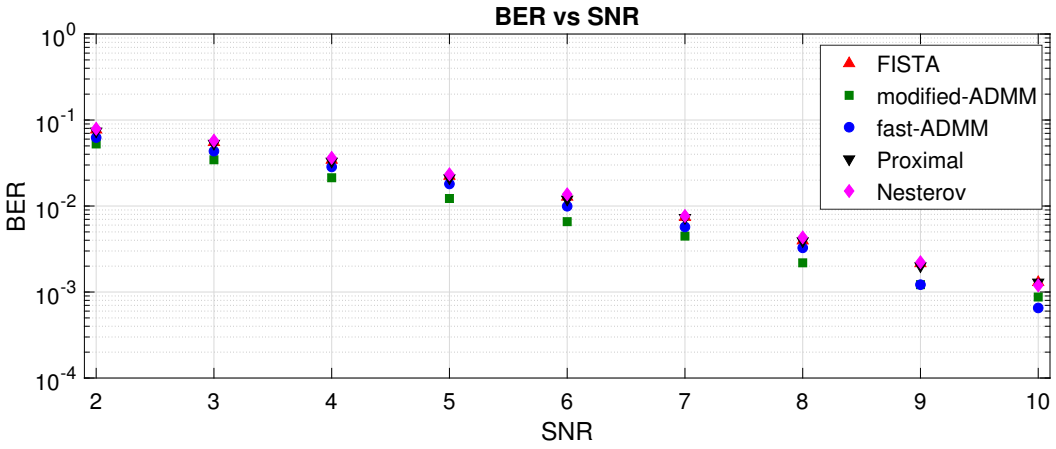

Figure 6: BER vs SNR without regularization update

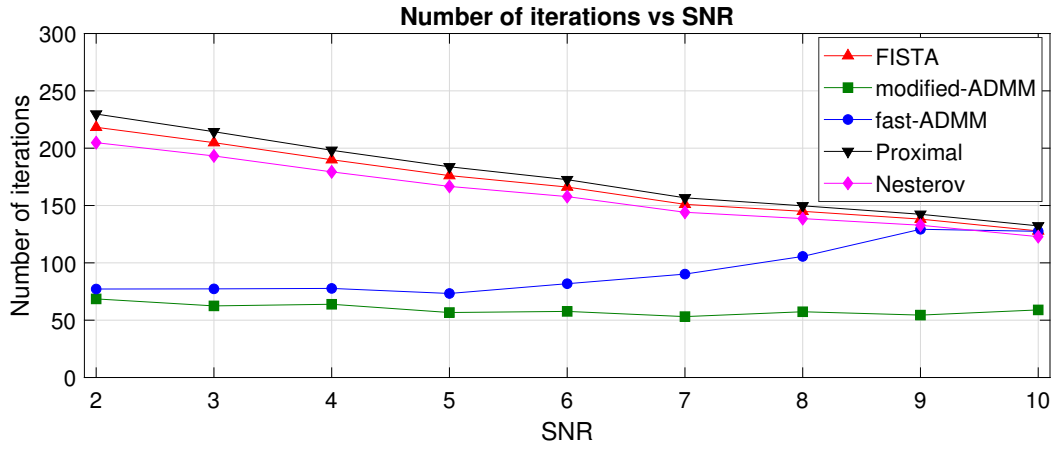

Figure 7: Number of iterations for convergence vs SNR

In figure 7, we analyze the number of iterations for convergence with different SNR value's. We observe that "modified-ADMM" converges very fast in comparison to other algorithms.

"fast-ADMM" converges faster for smaller values of SNR, but deteriorates with higher values of SNR. Among "proximal gradient" , "FISTA" and "Nesterov", we notice that "Nesterov" performs slightly better than "FISTA" which in turn performs better than "Proximal gradient" method. For higher values of SNR, the proposed "fast-ADMM" performs worse than "Proximal gradient", this could be due to higher number of restarts in the algorithm. Lastly, it's interesting to note that all the proxiaml based methods have only a slight variation in the number of iterations, this could be because of the structure of the problem (8) (As already pointed out in Section 5, the objective changes with every iteration, hence the behaviour of the methods are a little hard to analyze).

Next, we analyze the performance of all the algorithms by comparing average run-time with SNR in Table 6. As expected from Figure 7, "modified-ADMM" gives the best performance of all the algorithms and the performance of "fast-ADMM" deteriorates with higher values of SNR. We observe that for very high values of SNR, the average run-time of "Proximal gradient" method, "Nesterov" and "FISTA" are similar to "modified-ADMM". This could be attributed to the fact that a single iteration in ADMM based methods is more expensive compared to the proximal gradient based methods.

SNR	FISTA	modified-ADMM	fast-ADMM	Proximal	Nesterov
2	3.3	1.9	2.0	3.5	3.0
3	2.8	1.7	2.0	3.1	2.5
4	2.5	1.7	2.1	2.8	2.3
5	2.3	1.6	1.9	2.5	1.9
6	2.1	1.6	2.2	2.3	1.8
7	2.0	1.4	2.3	2.2	1.7
8	2.0	1.6	3.0	2.2	1.7
9	1.9	1.5	3.6	2.0	1.6
10	1.8	1.7	3.5	1.8	1.6

Table 6: Average CPU time in seconds

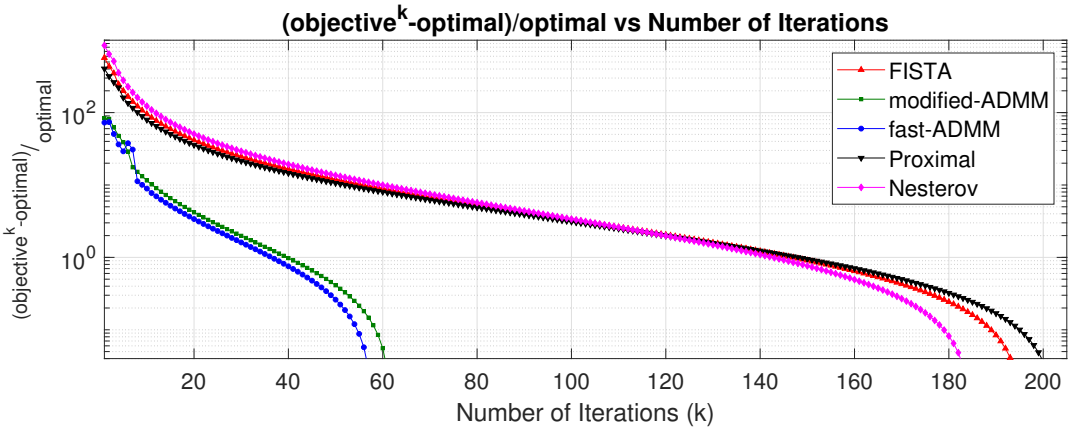

Figure 8: $\frac{objective\ value^{[k]} - optimal\ value}{optimal\ value}$ vs Number of Iterations

Next, we look into the variation of relative difference of objective with optimal value and the value of residuals with number of iterations in Figure 8 and Figure 9 respectively. As

expected for SNR = 5, "modified-ADMM" and "fast-ADMM" preform the best, they both converge in 60 iterations, whereas, "Nesterov", "FISTA" and "proximal gradient" take about 185, 192 and 203 iterations respectively.

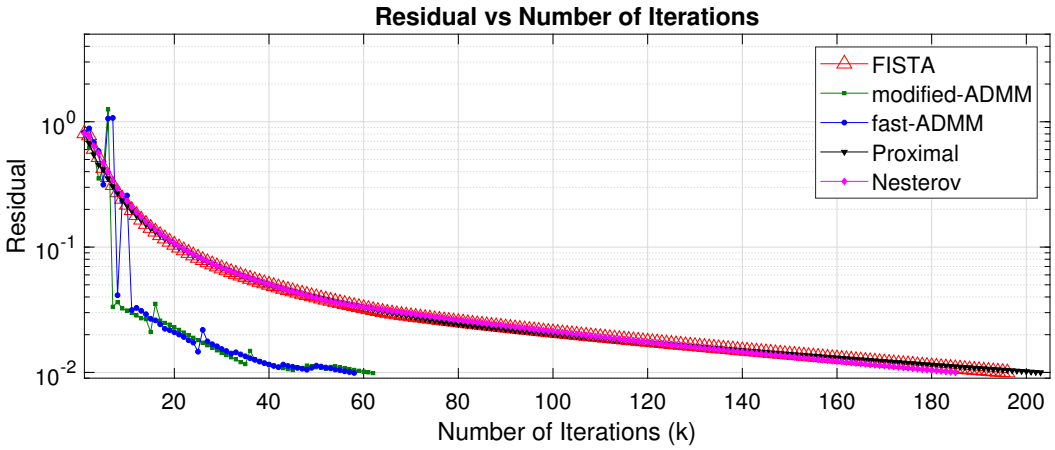

Figure 9: Residual vs Number of iterations

In Figure 9, only primal residuals of the ADMM based methods are used. We notice that the residuals of "modified ADMM" and "fast-ADMM" fluctuate and also drop down faster with the number of iterations compared to "proximal gradient", "Nesterov" and "FISTA". This could be because of the change in Lagrange parameter $\rho_t^{[k]}$ with number of iterations and enforced restarts in the ADMM based methods. Also, we note that the residuals of all the proximal gradient based methods, converge very slowly this could be because of the structure of the problem (8).

Comparing the run-times and number of iterations required for convergence from the last two subsections, we can conclude that regularization update α_t improves the performance of the algorithm. **Note:** It's not fair to compare both the models since the objective and tolerance values are different, but the key-point is that regularization of α_t as suggested in Section 4 is improving the performance.

7 Conclusion

In this report, we have considered the MUD performance for uplink grant-free NOMA systems and investigated optimization algorithms to jointly realize user activity and data detection. Simulation results for our model suggests that ADMM with varying penalty parameter performs best in comparison to other ADMM and proximal gradient based methods. Among proximal gradient based methods, "Nesterov" performs better than "FISTA" and "Proximal gradient". These conclusions are based on run-times, BER values, scaling factors and number of iterations required for convergence.

References

[1] L. Dai, B. Wang, Y. Yuan, S. Han, C. I, and Z. Wang. Non-orthogonal multiple access for 5g: solutions, challenges, opportunities, and future research trends. *IEEE Communications Magazine*, 53(9):74–81, Sep. 2015.

[2] Z. Ding, F. Adachi, and H. V. Poor. The application of mimo to non-orthogonal multiple access. *IEEE Transactions on Wireless Communications*, 15(1):537–552, Jan 2016.

[3] B. Shim and B. Song. Multiuser detection via compressive sensing. *IEEE Communications Letters*, 16(7):972–974, July 2012.

[4] Y. Wang and W. Yin. Sparse signal reconstruction via iterative support detection. *SIAM Journal on Imaging Sciences*, 3(3):462–491, 2010.

[5] A. C. Cirik, N. Mysore Balasubramanya, and L. Lampe. Multi-user detection using admm-based compressive sensing for uplink grant-free noma. *IEEE Wireless Communications Letters*, 7(1), Feb 2018.

[6] A. Beck. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.

[7] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Publishing Company, Incorporated, 1 edition, 2014.

[8] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014.

[9] W. Lu and N. Vaswani. Regularized modified bpdn for noisy sparse reconstruction with partial erroneous support and signal value knowledge. *IEEE Transactions on Signal Processing*, 60(1):182–196, Jan 2012.

[10] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[11] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences*, 7(3):1588–1623, 2014.