**FFTLasso: Large-Scale LASSO in the Fourier Domain**  
(Supplementary Material)

Adel Bibi, Hani Itani, Bernard Ghanem  
King Abdullah University of Science and Technology (KAUST), Saudi Arabia  
adel.bibi@kaust.edu.sa, bernard.ghanem@kaust.edu.sa

In this supplementary, we will discuss the details to the provided formulation in the main manuscript. Next, we will present an efficient matrix notation implementation of FFTLasso. Lastly, we discuss the memory efficiency of the algorithm. The original LASSO is given as follows:

$$\min_{c} \|Ac - b\|_2^2 + \lambda \|c\|_1$$  

(1)

Problem 1 is equivalent to the FFTLasso reformulation:

$$\min_{c} \|\tilde{A}c - b\|_2^2 + \lambda \|c\|_1 \quad \text{s.t} \quad Dc = 0,$$  

(2)

where $\tilde{A} \in \mathbb{R}^{m \times mn}$ and is block wise circulant. First and for reasons discussed in the main manuscript, we need to find the dual equivalence to problem 2. To do so, we introduce a new variable $r = \tilde{A}c - b$ as follows:

$$\min_{\tilde{c}, r} \|r\|_2^2 + \lambda \|\tilde{c}\|_1 \quad \text{s.t} \quad Dr = 0, \quad \tilde{A}c - b = r$$  

(3)

Then, the corresponding lagrangian is defined as follows:

$$\mathcal{L}(r, \tilde{c}, \Psi, \theta) := \|r\|_2^2 + \lambda \|\tilde{c}\|_1 + \Psi^T(\tilde{A}c - b - r) + \theta^T D\tilde{c}$$  

(4)

To find the dual problem, we minimize over the primal variables as follows:

$$\min_{r} \left[ \|r\|_2^2 - \Psi^H r \right] - \Psi^H b +$$

$$\min_{\tilde{c}} \left[ \lambda \|\tilde{c}\|_1 + \Psi^H \tilde{A}c + \theta^H D\tilde{c} \right]$$  

(5)

$$= -\Psi^H b - \frac{1}{4} \|\Psi\|_2^2 -$$

$$\lambda \max_{\tilde{c}} \left( \frac{\tilde{c}^H(\tilde{A}^H \Psi - D^H \theta)}{\lambda} - \|\tilde{c}\|_1 \right)$$  

(6)

$$= -\Psi^H b - \frac{1}{4} \|\Psi\|_2^2 - \frac{1}{\lambda} \tilde{A}^H \Psi - D^H \theta \|\tilde{c}\|_1 \leq 1$$  

(7)

Therefore, the dual problem is given as follows:

$$\min_{\Psi, \theta} \Psi^H b + \frac{1}{4} \|\Psi\|_2^2 \quad \text{s.t} \quad \tilde{A}^H \Psi + D^H \theta \|\tilde{c}\|_1 \leq \lambda$$  

(8)

Now, we apply ADMM to problem 8. First, we introduce the change of variables $\zeta = \tilde{A}^T \Psi + D^T \theta$ and append the $\|.\|_2$ constraint as an indicator function. Problem 8 is then equivalent to:

$$\min_{\Psi, \theta, \zeta} \frac{1}{4} \|\Psi\|_2^2 + \Psi^H b + \frac{1}{\lambda} \|\zeta\|_1 \leq \lambda$$  

$$\text{s.t} \quad \zeta = \tilde{A}^T \Psi + D^T \theta$$  

(9)

The augmented lagrangian is defined as follows:

$$\mathcal{L}(\Psi, \theta, \zeta, \tilde{c}) := \frac{1}{4} \|\Psi\|_2^2 + \Psi^H b + \frac{1}{\lambda} \|\zeta\|_1 \leq \lambda +$$

$$\tilde{c}^H(\tilde{A}^T \Psi + D^T \theta - \zeta) + \frac{\rho}{2} \|\tilde{A}^H \Psi + D^H \theta - \zeta\|_2^2$$  

(10)

The primal-dual update variables are given as follows: Update $\Psi$:

$$\Psi^{k+1} = \arg\min_{\Psi} \frac{1}{4} \|\Psi\|_2^2 + \Psi^H b + \tilde{c}^H \tilde{A}^H \Psi +$$

$$+ \frac{\rho}{2} \|\tilde{A}^H \Psi + D^H \theta - \zeta\|_2^2$$  

(11)

The previous objective minimization reduces to solving the following linear system:

$$(\rho \tilde{A} \tilde{A}^T + \frac{1}{2} I_m) \Psi^{k+1} = \tilde{A}(\rho \zeta - \rho D^T \theta - \tilde{c}) - b \quad (12)$$

Since $\tilde{A} e = \sum_i A_i e_i = F \sum_i \tilde{a}_i \odot \tilde{e}_i$ where $e = \rho \zeta - \rho D^T \theta - \tilde{c}$. The solution to the linear system is given as follows:

$$F(\rho \sum_i \text{diag}(\tilde{a}_i \odot \tilde{a}_i^*) + \frac{1}{2} I_m) F^H \Psi = F \sum_i \text{diag}(\tilde{a}_i^* \odot \tilde{e}_i^*) - b$$

By multiplying both sides by $F^H$ where $FF^H = I_m$, then:

$$\rho \sum_i \text{diag}(\tilde{a}_i \odot \tilde{a}_i^*) + \frac{1}{2} I_m) \Psi^* = \sum_i \text{diag}(\tilde{a}_i^* \odot \tilde{e}_i^*) - \tilde{b}^*$$

$$\Psi^* = \frac{\sum_i \tilde{a}_i^* \odot \tilde{e}_i^* - \tilde{b}^*}{\rho \sum_i \tilde{a}_i \odot \tilde{a}_i^* + \frac{1}{2}}$$  

(13)
Update $D^H \theta$:
\[
D^H \theta^{k+1} = \arg \min_{D^H \theta} \frac{1}{2} ||D^H \theta + \rho \hat{\Psi} + D^H \theta - \zeta||^2
\]
\[
D^H \theta^{k+1} = \zeta - \hat{c}/\rho - \hat{\Psi}
\]
(14)
To compute $\hat{\Psi}$ efficiently, the diagonalization trick can be applied again as follows:

\[
\hat{\Psi} = \begin{bmatrix} F(\hat{a}_1 \circ \Psi) \\ \vdots \\ F(\hat{a}_n \circ \Psi) \end{bmatrix}
\]
(15)

Update $\zeta$:
\[
\zeta^{k+1} = \arg \min_{\zeta} \frac{1}{2} ||\zeta - (\hat{\Psi} + D^T \theta + \hat{c}/\rho)||^2
\]
where $t = \hat{\Psi} + D^T \theta + \hat{c}/\rho$. Lastly, a Nesterov dual ascent is applied to the dual variables as follows:
Update $c$:
\[
\hat{c}_{k+1} = \hat{y}_k + \hat{\Psi} + D^H \theta - \zeta
\]
(18)
\[
\hat{y}_{k+1} = (1 + q)\hat{c}_{k+1} - q\hat{c}_k
\]
(19)
The algorithm is given as follows:

**Algorithm 1:** FFTLasso for Problem (9)

**Input:** $b, A, \hat{A}^*, E_1 = Y_1 = E_1 = T_1 = Z_1 = \Theta_1 = 0_{m \times n}, \Psi, \rho, \gamma > 1, q$

**Output:** $c$

while not converged do

**compute:** $E_{k+1} = \rho_k Z_k - \rho_k \Theta_k - C_k$

**update:** $\hat{\Psi}_{k+1} = \left( (\hat{\Psi} + E_{k+1,2}) - \hat{\Psi} \right) / \left( \rho_k \text{corr}^{aa} + \frac{1}{2} \right)$

**compute:** $\hat{A}^n \Psi_{k+1}$, see Eq (15)

**update:** $(\hat{\Psi} + D^T \theta_k + \hat{c}/\rho) \zeta_{k+1} = \text{sign}(t_k+1) \odot \min(\langle t_k+1, \lambda \rangle)$

**compute:** $\hat{c}_{k+1} = \hat{y}_k + \hat{\Psi}_{k+1} + D^H \theta_k - \zeta_{k+1}$

**compute:** $\hat{y}_{k+1} = (1 + q)\hat{c}_{k+1} - q\hat{c}_k$

$\rho_{k+1} = \gamma \rho_k$

end

c ← $C(1:m:end)$

Next we present a more efficient implementation to FFTLasso algorithm using matrix notation. First we define the new variables in matrix notation as follows: $e \in \mathbb{R}^{mn}$ is changed to $E \in \mathbb{R}^{m \times n}$, $D^H \theta$ is changed to $\Theta \in \mathbb{R}^{m \times n}$, $t \in \mathbb{R}^{mn}$ is changed to $T \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^{mn}$ is changed to $C \in \mathbb{R}^{m \times n}$, $y \in \mathbb{R}^{mn}$ is changed to $Y \in \mathbb{R}^{m \times n}$, and lastly $\zeta \in \mathbb{R}^{mn}$ is changed to $Z \in \mathbb{R}^{m \times n}$. The algorithm is then given as follows: In this matrix notation, there is no need

**Algorithm 2:** FFTLasso for Problem (9)

**Input:** $b, A, \hat{A}^*, F_m A, C_1 = Y_1 = E_1 = T_1 = Z_1 = \Theta_1 = 0_{m \times n}, \Psi, \rho, \gamma > 1, q$

**Output:** $c$

while not converged do

compute: $E_{k+1} = \rho_k Z_k - \rho_k \Theta_k - C_k$

update: $\hat{\Psi}_{k+1} = \left( (\hat{\Psi} + E_{k+1,2}) - \hat{\Psi} \right) / \left( \rho_k \text{corr}^{aa} + \frac{1}{2} \right)$

compute: $\hat{A}^n \Psi_{k+1}$, see Eq (15)

update: $(\Theta_{k+1} = (Z - \frac{1}{\rho_k} C_k - \hat{A}^n \Psi_{k+1})$

compute: $\hat{A}^n \Psi_{k+1}$, see Eq (15)

update: $(\Theta_{k+1} = (Z - \frac{1}{\rho_k} C_k - \hat{A}^n \Psi_{k+1})$

compute: $C_{k+1} = Y_k + \rho_k (\hat{A}^n \Psi_{k+1} + \Theta_{k+1} - Z_{k+1})$

compute: $Y_{k+1} = (1 + q)C_{k+1} - qC_k$

$\rho_{k+1} = \gamma \rho_k$

end

c ← $C(1:m:end)$

to reshape any of the variables for any operation unlike the case in Algorithm 1.

Regarding the memory efficiency, often there are overhead in computing the FFT of many long vectors that is when both $(m, n)$ are large. For instance, computing $F_m E$ for large $(m, n)$. The overhead of storage due to replicating $E$ is not always done efficiently. To circumvent this issue, we simply split the matrix $E$ into two pieces horizontally $E_1, E_2 \in \mathbb{R}^{n \times n}$ into an even and odd fashion similar to the butterfly algorithm used to compute the FFTs. Now we only compute the FFT of smaller vectors $\mathbb{C}^2 \mathbb{C}$ where we can simply combine them later. For completeness we show here how the procedure is carried out.

Let $v \in \mathbb{R}^n$, it can be written as follows $v = v_1 + v_2$ where $v_1 = [v0 0 v2 0 v3 \ldots]^H$ and where $v_2 = [0 v1 0 v3 \ldots]^H$. It is obvious that $\hat{v} = \hat{v}_1 + \hat{v}_2$ where $\hat{v}$ is the fourier transform of $v$. All is left is to com-
Figure 1. Runtime comparison between FFTLasso-GPU (on TitanX) and DL-ADMM (multi-core) at different CPU frequencies for square dictionaries $(m = n)$.

That means $	ilde{v}_1 = [\tilde{v}_e \tilde{v}_o]^H$ which is only a repeated version of the FFT of $v^2$ of the even part of $v$ and where $v_e = [v_0 \ v_2 \ ...]$. A similar trick can be applied to $v_2$ followed by a shift to the right. One can easily show that $	ilde{v}_1 = [\tilde{v}_e \tilde{v}_o]^H \otimes p$ where $p(i) = \exp(-j\pi ki/m) \forall i = 0, 1, ..., m$. Therefore,

$$\tilde{v} = [\tilde{v}_e^H \tilde{v}_o^H]^H + [\tilde{v}_e^H \tilde{v}_o^H]^H \otimes p$$

Lastly, in here we present the results of both the TitanX (1.0GHz/core) and TitanX Pascal (1.4GHz/core) experiments comparing FFTLasso to different multi-core CPU frequencies of DL-ADMM. Figure 1. shows a comparison between several hardware (GPUs) of FFTLasso against DL-ADMM on high frequency multi core CPUs. It is clear that depending on the hardware used and the dictionary $A$ used one would prefer using DL-ADMM over FFTLasso and vice versa. However, figure 1 clearly shows incremental monotone improvement in the speed of FFTLasso as the dictionary size grows large. Specifically, FFTLasso is mostly optimized for any dictionary that is of powers of an integer radix (in this case radix 2).