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

Adel Bibi, Hani Itani, and Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Saudi Arabia
adel.bibi@kaust.edu.sa, hmi10@mail.aub.edu, 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:

$$\min_{\mathbf{c}} ||\mathbf{A}\mathbf{c}-\mathbf{b}||_2^2+\lambda||\mathbf{c}||_1 \tag{1}$$

Problem 1 is equivalent to the FFTLasso reformulation:

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

where $\tilde{\mathbf{A}} \in \mathbb{R}^{m\times mn}$ 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 $\mathbf{r}=\tilde{\mathbf{A}}\tilde{\mathbf{c}}-\mathbf{b}$ as follows:

$$\min_{\tilde{\mathbf{c}},\mathbf{r}} \ \|\mathbf{r}\|_2^2+\lambda\|\tilde{\mathbf{c}}\|_1 \quad \text{s.t} \quad \mathbf{D}\tilde{\mathbf{c}}=\mathbf{0}, \ \ \tilde{\mathbf{A}}\tilde{\mathbf{c}}-\mathbf{b}=\mathbf{r} \tag{3}$$

Then, the corresponding lagrangian is defined as follows:

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

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

$$\min_{\mathbf{r}}\Big[||\mathbf{r}||_2^2-\Psi^\top\mathbf{r}\Big]-\Psi^\top\mathbf{b}+\min_{\tilde{\mathbf{c}}}\Big[\lambda||\tilde{\mathbf{c}}||_1+\Psi^\top\tilde{\mathbf{A}}\tilde{\mathbf{c}}+\theta^\top\mathbf{D}\tilde{\mathbf{c}}\Big] \tag{4}$$

$$=-\Psi^\top\mathbf{b}-\frac{1}{4}\|\Psi\|_2^2-\lambda\max_{\tilde{\mathbf{c}}}\Big(\frac{\tilde{\mathbf{c}}^\top(-\tilde{\mathbf{A}}^\top\Psi-\mathbf{D}^\top\theta)-\|\tilde{\mathbf{c}}\|_1}{\lambda}\Big) \tag{5}$$

$$=-\Psi^\top\mathbf{b}-\frac{1}{4}\|\Psi\|_2^2-\mathbb{1}_{\frac{1}{\lambda}\|-\tilde{\mathbf{A}}^\top\Psi-\mathbf{D}^\top\theta\|_\infty\leqslant 1} \tag{6}$$

Therefore, the dual problem is given as follows:

$$\min_{\Psi,\theta}\Psi^\top\mathbf{b}+\frac{1}{4}\|\Psi\|_2^2 \quad \text{s.t} \quad \|\tilde{\mathbf{A}}^\top\Psi+\mathbf{D}^\top\theta\|_\infty\leqslant\lambda \tag{7}$$

Now, we apply ADMM to problem 7. First, we introduce the change of variables $\zeta=\tilde{\mathbf{A}}^T\Psi+\mathbf{D}^T\theta$ and append the $\|.\|_\infty$ constraint as an indicator function. Problem 7 is then equivalent to:

$$\begin{aligned}&\min_{\Psi,\theta,\zeta}\frac{1}{4}\|\Psi\|_2^2+\Psi^\top\mathbf{b}+\mathbb{1}_{\|\zeta\|_\infty\leqslant\lambda}\\ &\text{s.t} \ \ \zeta=\tilde{\mathbf{A}}^T\Psi+\mathbf{D}^T\theta\end{aligned} \tag{8}$$

The augmented lagrangian is defined as follows:

$$\begin{aligned}&\mathcal{L}(\Psi,\theta,\zeta,\tilde{\mathbf{c}})=\frac{1}{4}\|\Psi\|_2^2+\Psi^\top\mathbf{b}+\mathbb{1}_{\|\zeta\|_\infty\leqslant\lambda}+\\ &\tilde{\mathbf{c}}^\top(\tilde{\mathbf{A}}^T\Psi+\mathbf{D}^T\theta-\zeta)+\frac{\rho}{2}||\tilde{\mathbf{A}}^\top\Psi+\mathbf{D}^T\theta-\zeta||_2^2\end{aligned} \tag{9}$$

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

$$\begin{aligned}\Psi^{k+1}=\text{argmin}_\Psi \ \ &\frac{1}{4}\|\Psi\|_2^2+\Psi^\top\mathbf{b}+\tilde{\mathbf{c}}^\top\tilde{\mathbf{A}}^\top\Psi\\ &+\frac{\rho}{2}||\tilde{\mathbf{A}}^\top\Psi+\mathbf{D}^\top\theta-\zeta||_2^2\end{aligned} \tag{10}$$

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

$$(\rho\tilde{\mathbf{A}}\tilde{\mathbf{A}}^T+\frac{1}{2}\mathbf{I}_m)\Psi^{k+1}=\tilde{\mathbf{A}}(\rho\zeta-\rho\mathbf{D}^T\theta-\tilde{\mathbf{c}})-\mathbf{b} \tag{11}$$

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

$$\mathbf{F}(\rho\sum_i^N diag(\hat{\mathbf{a}}_i\odot\hat{\mathbf{a}}_i^*)+\frac{1}{2}\mathbf{I}_m)\mathbf{F}^H\Psi=\mathbf{F}\sum_i^N diag(\hat{\mathbf{a}}_i^*\odot\hat{\mathbf{e}}_i^*)-\mathbf{b}$$

By multiplying both sides by $\mathbf{F}^H$ where $\mathbf{F}\mathbf{F}^H=\mathbf{I}_m$, then:

$$(\rho\sum_i^N diag(\hat{\mathbf{a}}_i\odot\hat{\mathbf{a}}_i^*)+\frac{1}{2}\mathbf{I}_m)\hat{\Psi}^*=\sum_i^N diag(\hat{\mathbf{a}}_i^*\odot\hat{\mathbf{e}}_i^*)-\hat{\mathbf{b}}^*$$

$$\hat{\Psi}^*=\frac{\sum_i^N\hat{\mathbf{a}}_i^*\odot\hat{\mathbf{e}}_i^*-\hat{\mathbf{b}}^*}{\rho\sum_i^N\hat{\mathbf{a}}_i\odot\hat{\mathbf{a}}_i^*+\frac{1}{2}} \tag{12}$$

**Update $\mathbf{D}^\top\theta$:**

$$\begin{aligned}\mathbf{D}^\top\theta^{k+1} &= \text{argmin}_{\mathbf{D}^\top\theta}\ \tilde{\mathbf{c}}^\top\mathbf{D}^\top\theta + \frac{\rho}{2}\|\tilde{\mathbf{A}}^\top\Psi + \mathbf{D}^\top\theta - \zeta\|_2^2 \\ \mathbf{D}^\top\theta^{k+1} &= \zeta - \tilde{\mathbf{c}}/\rho - \tilde{\mathbf{A}}^\top\Psi\end{aligned} \tag{13}$$

To compute $\tilde{\mathbf{A}}^\top\Psi$ efficiently, the diagonalization trick can be applied again as follows:

$$\tilde{\mathbf{A}}^\top\Psi = \begin{bmatrix}\mathbf{F}(\hat{\mathbf{a}}_1 \odot \Psi^*) \\ \vdots \\ \mathbf{F}(\hat{\mathbf{a}}_n \odot \Psi^*)\end{bmatrix} \tag{14}$$

**Update $\zeta$:**

$$\begin{aligned}\zeta^{k+1} =& \text{argmin}_\zeta\ \ \mathbb{1}_{\|\zeta\|_\infty \leqslant \lambda} - \tilde{\mathbf{c}}^\top\zeta + \frac{\rho}{2}||\tilde{\mathbf{A}}^T\Psi + \mathbf{D}^T\theta - \zeta||_2^2 \\ =& \text{argmin}_\zeta\ \ ||\zeta - (\tilde{\mathbf{A}}^\top\Psi + \mathbf{D}^T\theta + \tilde{\mathbf{c}}/\rho)||_2^2 \\ & \text{s.t. } ||\zeta||_\infty \leqslant \lambda\end{aligned} \tag{15}$$

The former is just a simple Euclidian projection to the $\|.\|_\infty$ ball and is given as follows:

$$\zeta^{k+1} = \text{sign}(\mathbf{t}) \odot \min(|\mathbf{t}|, \lambda) \tag{16}$$

where $\mathbf{t} = \tilde{\mathbf{A}}^T\Psi + \mathbf{D}^T\theta + \tilde{\mathbf{c}}/\rho$. Lastly, a Nesterov dual ascent is applied to the dual variables as follows:

**Update $\tilde{\mathbf{c}}$:**

$$\tilde{\mathbf{c}}_{k+1} = \tilde{\mathbf{y}}_k + \rho(\tilde{\mathbf{A}}^\top\Psi + \mathbf{D}^\top\theta - \zeta) \tag{17}$$

$$\tilde{\mathbf{y}}_{k+1} = (1+q)\tilde{\mathbf{c}}_{k+1} - q\tilde{\mathbf{c}}_k \tag{18}$$

The algorithm is given as follows:

**Algorithm 1:** FFTLasso for Solving Problem (8)

**Input** : $\mathbf{b}, \mathbf{A}, \tilde{\mathbf{c}}_1 = \tilde{\mathbf{y}}_1 = \mathbf{e}_1 = \mathbf{t}_1 = \zeta_1 = \mathbf{D}^\top\theta_1 = \mathbf{0}_{mn}, \Psi = \mathbf{0}_m, \lambda, \rho_1, \gamma > 1, q.$

**Output**: $\mathbf{c}$

**while** not converged **do**

- **compute:** $\mathbf{e}_{k+1} = \rho_k\zeta_k - \rho_k\mathbf{D}^\top\theta_k - \tilde{\mathbf{c}}_k$
- $\hat{\Psi}^*$ **update:** $\hat{\Psi}^*_{k+1} = \frac{\sum_i^N \hat{\mathbf{a}}_i^* \odot \hat{\mathbf{e}}^*_{ik+1} - \hat{\mathbf{b}}^*}{\rho_k \sum_i^N \hat{\mathbf{a}}_i \odot \hat{\mathbf{a}}_i^* + \frac{1}{2}}$
- **compute:** $\tilde{\mathbf{A}}^\top\Psi_{k+1}$, see Eq (14)
- $\mathbf{D}^\top\theta$ **update:**
- $(\mathbf{D}^\top\theta_{k+1}) = (\zeta_k - \frac{1}{\rho_k}\tilde{\mathbf{c}}_k - \tilde{\mathbf{A}}^\top\Psi_{k+1})$
- $(\mathbf{D}^\top\theta_{k+1})_{\text{1:m:end}} = \mathbf{0}_n$
- **compute:** $\mathbf{t}_{k+1} = \tilde{\mathbf{A}}^\top\Psi_{k+1} + \mathbf{D}^\top\theta_{k+1} + \tilde{\mathbf{c}}_k/\rho_k$
- $\zeta$ **update:** $\zeta_{k+1} = \text{sign}(\mathbf{t}_{k+1}) \odot \min(|\mathbf{t}_{k+1}|, \lambda)$
- $\tilde{\mathbf{c}}$ **update:**
- $\tilde{\mathbf{c}}_{k+1} = \tilde{\mathbf{y}}_k + \rho_k(\tilde{\mathbf{A}}^\top\Psi_{k+1} + \mathbf{D}^\top\theta_{k+1} - \zeta_{k+1})$
- **compute:** $\tilde{\mathbf{y}}_{k+1} = (1+q)\tilde{\mathbf{c}}_{k+1} - q\tilde{\mathbf{c}}_k$
- $\rho_{k+1} = \gamma\rho_k$

**end**

$\mathbf{c} \leftarrow \tilde{\mathbf{c}}(1{:}m{:}\text{end})$

Next we present a more efficient implementation to FFTLasso algorithm using matrix notation. First we define the new variables in matrix notation as follows: $\mathbf{e} \in \mathbb{R}^{mn}$ is changed to $\mathbf{E} \in \mathbb{R}^{m\times n}$, $\mathbf{D}^\top\theta$ is changed to $\Theta \in \mathbb{R}^{m\times n}$, $\mathbf{t} \in \mathbb{R}^{mn}$ is changed to $\mathbf{T} \in \mathbb{R}^{m\times n}$, $\mathbf{c} \in \mathbb{R}^{mn}$ is changed to $\mathbf{C} \in \mathbf{R}^{m\times n}$, $\mathbf{y} \in \mathbb{R}^{mn}$ is changed to $\mathbf{Y} \in \mathbb{R}^{m\times n}$, and lastly $\zeta \in \mathbb{R}^{mn}$ is changed to $\mathbf{Z} \in \mathbb{R}^{m\times n}$. The algorithm is then given as follows: In this matrix notation, there is no need to reshape any of the variables for any operation unlike the case in Algorithm 1.

**Algorithm 2:** FFTLasso for Solving Problem (8)

**Input** : $\mathbf{b}, \mathbf{A}, \mathbf{A}^* = \mathbf{F}_m\mathbf{A}, \tilde{\mathbf{C}}_1 = \tilde{\mathbf{Y}}_1 = \mathbf{E}_1 = \mathbf{T}_1 = \mathbf{Z}_1 = \Theta_1 = \mathbf{0}_{m\times n} \in \mathbb{R}^{m\times n}, \Psi = \mathbf{0}_m, \mathbf{corr}^{aa} = \text{sum}(\hat{\mathbf{A}}^* \odot \hat{\mathbf{A}}^*,2), \lambda, \rho_1, \gamma > 1, q.$

**Output**: $\mathbf{c}$

**while** not converged **do**

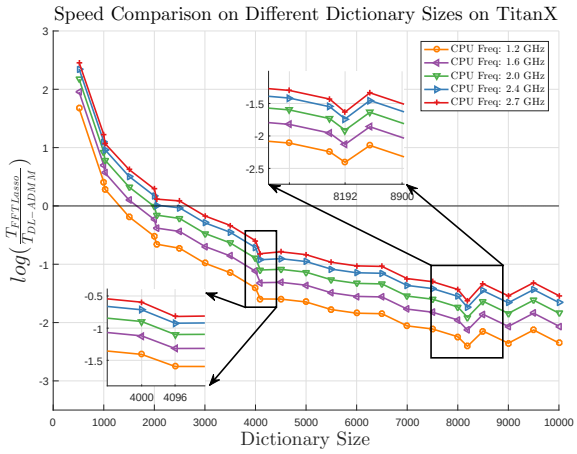
- **compute:** $\mathbf{E}_{k+1} = \rho_k\mathbf{Z}_k - \rho_k\Theta_k - \tilde{\mathbf{C}}_k$
- $\hat{\Psi}^*$ **update:** $\hat{\Psi}^*_{k+1} = \left(\text{sum}(\hat{\mathbf{A}}^* \odot \tilde{\mathbf{E}}^*,2) - \hat{\mathbf{b}}^*\right) / \left(\rho_k\mathbf{corr}^{aa} + \frac{1}{2}\right)$
- **compute:** $\tilde{\mathbf{A}}^\top\Psi_{k+1}$, see Eq (14)
- $\Theta$ **update:** $(\Theta_{k+1}) = (\mathbf{Z} - \frac{1}{\rho_k}\tilde{\mathbf{C}}_k - \tilde{\mathbf{A}}^\top\Psi_{k+1})$
- $\Theta_{(\text{1:end,:})} = \mathbf{0}_n^\top$
- **compute:** $\mathbf{T}_{k+1} = \tilde{\mathbf{A}}^\top\Psi_{k+1} + \Theta_{k+1} + \tilde{\mathbf{c}}_k/\rho_k$
- $\mathbf{Z}$ **update:** $\mathbf{Z}_{k+1} = \text{sign}(\mathbf{T}_{k+1}) \odot \min(|\mathbf{T}_{k+1}|, \lambda)$
- $\tilde{\mathbf{C}}$ **update:**
- $\tilde{\mathbf{C}}_{k+1} = \tilde{\mathbf{Y}}_k + \rho_k(\tilde{\mathbf{A}}^\top\Psi_{k+1} + \Theta_{k+1} - \mathbf{Z}_{k+1})$
- **compute:** $\tilde{\mathbf{Y}}_{k+1} = (1+q)\tilde{\mathbf{C}}_{k+1} - q\tilde{\mathbf{C}}_k$
- $\rho_{k+1} = \gamma\rho_k$

**end**

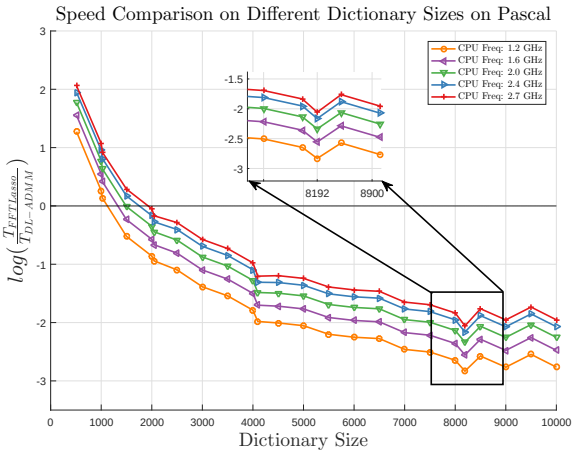
$\mathbf{c} \leftarrow \tilde{\mathbf{C}}(1{:}\text{end},:)$

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 $\mathbf{FE}$ for large (m,n). The overhead of storage due to replicating $\mathbf{E}$ is not always done efficiently. To circumvent this issue, we simply split the matrix $\mathbf{E}$ into two pieces horizontally $\mathbf{E}_1, \mathbf{E}_2 \in \mathbb{R}^{\frac{m}{2}\times n}$ 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 $\frac{m}{2}$ where we can simply combine them later. For completeness we show here how the procedure is carried out.

Let $\mathbf{v} \in \mathbf{R}^m$, it can be written as follows $\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2$ where $\mathbf{v}_1 = \begin{bmatrix}v0 & 0 & v2 & \dots\end{bmatrix}^\top$ and where $\mathbf{v}_2 = \begin{bmatrix}0 & v1 & 0 & v3 & \dots\end{bmatrix}^\top$. It is obvious that $\tilde{\mathbf{v}} = \tilde{\mathbf{v}}_1 + \tilde{\mathbf{v}}_2$ where $\tilde{\mathbf{v}}$ is the fourier transform of $\mathbf{v}$. All is left is to com-




(a) $m = n$.




(b) $m = n$.

Figure 1. Runtime comparison between FFTLasso-GPU (on TitanX) and DL-ADMM (multi-core) at different CPU frequencies for square dictionaries ($m = n$)

pute both $\tilde{\mathbf{v}}_1$, and $\tilde{\mathbf{v}}_2$ using lower order FFT as follows:

$$\begin{aligned}\mathbf{F}\mathbf{v}_1 &= \sum_{i=0}^{m} \mathbf{v}_1(i) \exp \frac{-j2\pi ki}{m} = \sum_{0,2,4,..}^{m} \mathbf{v}_1(i) \exp \frac{-j2\pi ki}{m} \\ &= \sum_{i=0,2,4,..}^{m} \mathbf{v}_e(\frac{i}{2}) \exp \frac{-j2\pi ki}{m} = \sum_{v=0,1,...}^{\frac{m}{2}} \mathbf{v}_e(i) \exp \frac{-j2\pi ki}{\frac{m}{2}} \\ &= \tilde{\mathbf{v}}_e(i) \forall i = 0, 1, ..., m\end{aligned}$$

That means $\tilde{\mathbf{v}}_1 = \begin{bmatrix}\tilde{\mathbf{v}}_e & \tilde{\mathbf{v}}_e\end{bmatrix}^\top$ which is only a repeated version of the FFT of $\frac{m}{2}$ of the even part of $\mathbf{v}$ and where $\mathbf{v}_e = \begin{bmatrix}v0 & v2 & \dots\end{bmatrix}$. A similar trick can be applied to $\mathbf{v}_2$ followed by a shift to the right. One can easily show that $\tilde{\mathbf{v}}_1 = \begin{bmatrix}\tilde{\mathbf{v}}_e & \tilde{\mathbf{v}}_e\end{bmatrix}^\top \odot \mathbf{p}$ where $\mathbf{p}(i) = \exp \frac{-j2\pi ki}{m} \quad \forall i = 0, 1, ..., m$. Therefore,

$$\tilde{\mathbf{v}} = \begin{bmatrix}\tilde{\mathbf{v}}_e^\top & \tilde{\mathbf{v}}_e^\top\end{bmatrix}^\top + \begin{bmatrix}\tilde{\mathbf{v}}_o^\top & \tilde{\mathbf{v}}_o^\top\end{bmatrix}^\top \odot \mathbf{p}$$

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 $\mathbf{A}$ 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).