Using the Alternating Direction of Multipliers Method for the shortest vector in a lattice

Wissam AlKendi, Stéphane Chrétien and Christophe Guyeux

Abstract

The Lenstra-Lenstra-Lovász (LLL) algorithm has been widely used for finding the shortest vector in a lattice, with applications in cryptography, coding theory, and quantum computing. However, LLL has a worst-case exponential running time, and its practical performance depends heavily on the lattice structure. In this paper, we propose a new algorithm for finding the shortest vector in a lattice that outperforms LLL. Our algorithm is a stochastic version of the ADMM algorithm proposed by Takapoui, Moehle, Boyd and Bemporad. We perform extensive experiments on various lattice structures and show that our algorithm significantly improves the performance of LLL in terms of running time and solution quality. The results demonstrate the effectiveness and efficiency of our approach, and provide new insights into the shortest vector problem.

I. Introduction

Constructing nearly orthogonal bases for lattices is an important problem in mathematics with multiple practical applications such as cryptography [4]. The next figure shows an exemple of a two dimensional lattice and two different bases for this lattice, the red one being more "orthogonal" than the black one.

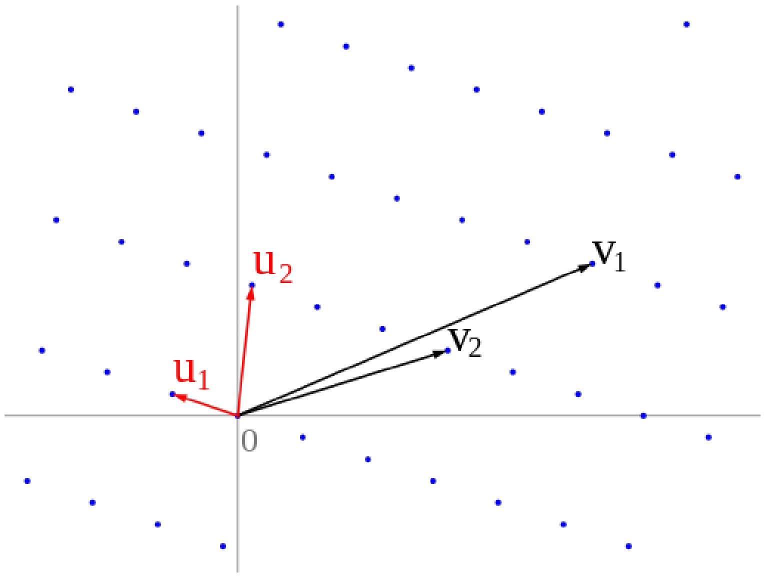

Fig. 1: Two lattice bases, the red one being more "orthogonal" than the black one

The Lenstra-Lenstra-Lovasz algorithm proposed by [3] is a central tool for finding near orthogonal bases. The goal of the present work is the present an alternative approach to the LLL algorithm based on recent techniques from optimisation theory, namely the alternating direction method of multipliers (ADMM). In particular, we investigate the numerical performance of the ADMM on random instances and show that is has superior performance in practice, as compared with the LLL algorithm, both for the problem of finding a short vector in the computed basis, and for the problem of designing fast and scalable algorithms.

II. Background on lattice reduction

We will use the following notations. The real vector space $\mathbb{R}^n, n \geq 1$ is provided with its Euclidean structure and the Lebesgue measure, denoted μ. The canonical basis of $\mathbb{R}^n$ is denoted by $(e_1, e_2, \ldots, e_n)$. The scalar product of $v, u \in \mathbb{R}^n$, and the Euclidean norm of u are respectively denoted by $v \cdot u$, and $\|u\| = (u \cdot u)^{1/2}$. To a part $E \subseteq \mathbb{R}^n$, we associate the real vector space generated by E, which we denote by$\langle E \rangle$.

The set of matrices with n rows and m columns, with coefficients in a set $\mathbb{S}$ (in practice $\mathbb{R}$, $\mathbb{Q}$ or $\mathbb{Z}$) is denoted $\mathbb{S}^{n\times m}$. For a matrix M, we denote its transpose by $M^\top$, its determinant by $\det M$ and its inverse matrix (when it exists) by M^{-1}.

The Gram-Schmidt orthogonalisation plays a central role for the problem of finding a short vector in a lattice, which is our main focus in the present paper. In the following section, we summarise the main concepts and results about the Gram-Schmidt process.

A. Gram-Schmidt orthogonalisation

To a system $B = (b_1, \ldots, b_p)$ of p vectors of $\mathbb{R}^n$, we associate the matrix whose rows are the vectors b_i expressed in the canonical basis $(e_1, e_2, \ldots, e_n)$ of $\mathbb{R}^n$. This matrix will be called the row matrix of $(b_1, \ldots, b_p)$ and will also be, with a slight abuse of language, designated by B. The sets $[\![a, b]\!]$ defined by $[\![a, b]\!] := [a, b] \cap \mathbb{Z}$ will be called integer intervals. The open (resp. closed) ball of radius ρ centered at a is denoted by $B(a, \rho)$ (resp. $\bar{B}(a, \rho)$) and is as usual defined by

$$B(a, \rho) = \{x \in \mathbb{R}^n \mid \|x - a\| < \rho\}, \quad \bar{B}(a, \rho) = \{x \in \mathbb{R}^n \mid \|x - a\| \leq \rho\}.$$

For the sake of completeness, we now give the definition of the well known Gram-Schmidt orthogonalisation.

Definition II-A.1. *(Gram-Schmidt orthogonalisation, GSO). Let $B = (b_1, \ldots, b_p)$ be a family of linearly independent vectors of $\mathbb{R}^n$. We denote by B_i the starting family, $B_i := (b_1, \ldots, b_i)$ and by H_i the $\mathbb{R}$-vector space generated by B_i. The Gram-Schmidt orthogonalized family is the orthogonal family $B^\star = \left(b_1^\star, \ldots, b_p^\star\right)$ formed by the vectors $b_i^\star$, where $b_i^\star$ is the orthogonal projection of b_i onto the orthogonal of H_{i-1}. More precisely*

$$\begin{aligned} b_1^\star &= b_1 \\ b_i^\star &= b_i - \sum_{k=1}^{i-1} m_{i,k} b_k^\star, \quad \text{with} \quad m_{i,j} = \frac{b_i \cdot b_j^\star}{\|b_j^\star\|^2} \quad \text{for} \quad 1 \leq j < i \leq p \end{aligned}$$

We also pose $m_{i,i} = 1$ for $1 \leq i \leq p$ and $m_{i,j} = 0$ for $1 \leq i < j \leq p$. This Gram-Schmidt orthogonalization procedure also constructs the matrix $\mathcal{P} \in \mathbb{R}^{p \times p}$ whose entry $m_{i,j}$ is defined above.

For simplicity, we will denote by B the matrix $\mathbb{R}^{p \times n}$ whose i^{th} row is the vector b_i. If $B^\star$ is the matrix $\mathbb{R}^{p \times n}$ whose i^{th} row is the vector $b_i^\star$, then we have $B = \mathcal{P} B^\star$.

Definition II-A.2. *Let $B = (b_1, \ldots, b_p)$ and let $B^\star = \left(b_1^\star, \ldots, b_p^\star\right)$ be the family obtained after the Gram-Schmidt orthogonalisation process. The ℓ_2-norm of $b_i^\star$, denoted by ℓ_i, is called the i^{th} Siegel length.*

Definition II-A.3. *(Gram matrix). The Gram matrix of a system $B = (b_1, \ldots, b_p)$ of p vectors of $\mathbb{R}^n$, denoted by $G(b_1, \ldots, b_p)$ or $G(B)$, is the matrix of $\mathbb{R}^{p \times p}$ defined by*

$$G_{ij} = b_i \cdot b_j \quad \forall i, j \in [\![1, p]\!].$$

If the rows of the matrix B are the vectors of the system $(b_1, \ldots, b_p)$, then the Gram matrix is written $G(B) = B \cdot B^\top$.

Then, we have

$$\det G(B) = \prod_{i=1}^{p} \|b_i^\star\|^2 = \prod_{i=1}^{p} \ell_i^2.$$

We now turn to the heart of the matter, namely the concepts of lattice and basis reduction.

B. Lattices

A Euclidean lattice of $\mathbb{R}^n$ is the set of linear combinations with integer coefficients of a family $\{b_1, \ldots, b_p\}$ of p linearly independent vectors of $\mathbb{R}^n$, called the lattice basis.

The following result is a basic rephrasing of the notion of lattice in terms of groups.

Proposition II-B.1. *The following assertions are equivalent:*

(i) *$\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^n$, which generates a vector subspace of dimension p.*
(ii) *There exists a system $B = \{b_1, \ldots, b_p\}$ of p linearly independent vectors of $\mathbb{R}^n$, for which*

$$\mathcal{L} = \left\{ \sum_{i=1}^{d} x_i b_i \mid \quad x_i \in \mathbb{Z} \quad \forall i \in [\![1, p]\!] \right\}.$$

The next result is fundamental for justifying the approach adopted in the LLL algorithm to use orthogonalised bases.

Proposition II-B.2. *Let $\mathcal{L}$ be a lattice generated by a basis $B = (b_1, \ldots, b_p)$. We denote by $B^\star$ the orthogonalized basis of B. Then, for all $w \in \mathcal{L} \backslash \{0\}$ we have*

$$\|w\| \geq \min\{\|b_i^\star\|; \quad i \in [\![1, p]\!]\}.$$

We now define key quantities about lattices that relate to the length of the vectors in the lattice.

Definition II-B.3. *The first minimum of the network $\mathcal{L}$, denoted by $\lambda_1(\mathcal{L})$, is the norm of a shortest nonzero vector of $\mathcal{L}$. More generally, the i-minimum of the $\mathcal{L}$ lattice, denoted by $\lambda_i(\mathcal{L})$, is the smallest positive real number ρ for which the closed ball of radius ρ centered at the origin contains at least i vectors linearly of the $\mathcal{L}$ lattice,*

$$\lambda_i(\mathcal{L}) := \min\{\rho > 0 \mid \dim(\bar{B}(0,\rho) \cap \mathcal{L}) \geq i\}.$$

The following theorem by Minkowski is of great relevance, since a condition is given on the minimum size of a set to contain at least one point in the network.

Theorem II-B.4. *(Minkowski). Consider a Euclidean lattice $\mathcal{L}$ of dimension p. We denote by μ the p-dimensional Lebesgue measure, and consider a subset C of the vector subspace generated by $\mathcal{L}$, μ-measurable. which is convex, symmetric about the origin, and verifies $\mu(C) > 2^p \det \mathcal{L}$. Then C contains at least one point of $\mathcal{L}$.*

Basis reduction is key to the construction of the LLL algorithm.

Definition II-B.5. *(Informal definition of the notion of reduced basis). A basis $B = (b_1, \ldots, b_p)$ formed of p vectors of $\mathbb{R}^n$ is a reduced basis if it is formed of short enough and orthogonal enough vectors. These criteria are measured quantitatively by a markup of the orthogonality defect $\rho(B)$ and the length defects $\theta_i(B)$ defined respectively by*

$$\rho(B) = \frac{1}{\det \mathcal{L}(B)} \prod_{i=1}^{p} \|b_i\| = \prod_{i=1}^{p} \frac{\|b_i\|}{\|b_i^\star\|}, \quad \theta_i(B) = \frac{\|b_i\|}{\lambda_i(\mathcal{L}(B))}$$

The orthogonality defect $\rho(B)$ is at least 1, with equality only when the basis B is orthogonal. As a lattice does not have an orthogonal basis, we have in general $\rho(B) > 1$. The basis is "fairly" orthogonal when its orthogonality defect $\rho(B)$ is increased. The length defects are also at least equal to 1. The equalities $\theta_i(B) = 1$ can only occur simultaneously when the basis is minimal, i.e., formed by vectors realizing successive minima. The existence of a minimal basis is not always guaranteed, as soon as the dimension p verifies $p \geq 5$.

Definition II-B.6. *(Reduction). Given a basis B, reducing B consists in finding an equivalent reduced basis.*

A good notion of reduction must establish a compromise between the quality of the reduced basis and the complexity of the reduction algorithm. Such a compromise is achieved by the LLL algorithm, invented by Lenstra, Lenstra and Lovász in 1982, presented in the next subsection.

C. Lovász reduced bases and the LLL algorithm

A basis $(a_1, \ldots, a_n)$ of a lattice L of rank n in $\mathbb{R}^p$ is called Lovász-reduced if the lengths l_i of the vectors $\hat{a}_i$, $i = 1, \ldots, n$ and the entries $m_{i,j}$ of the matrix m satisfy the following two conditions:

- properness:
$$|m_{i,j}| \leqslant \frac{1}{2} \text{ for } 1 \leqslant j < i \leqslant n.$$
- Lovász condition:
$$l_i^2 \leqslant s^2 \left(l_{i+1}^2 + m_{i+1,i}^2 l_i^2\right) \text{ for } 1 \leqslant i \leqslant n-1, \tag{1}$$

where s is a real parameter in the interval $]1, 2[$, that is usually chosen to be equal to $t > 2/\sqrt{3}$.

The LLL algorithm is defined below in Algorithm 1 and makes use of the integral part of the Gram-Schmidt coefficients.

The main result about the orthogonality defect of the LLL algorithm is the following theorem.

Theorem II-C.1. *Consider a real $s > 2/\sqrt{3}$. The LLL algorithm constructs from a basis $B := (b_1, \ldots, b_p)$ whose size $\tau(B)$ is defined as*

$$\tau(B) = \Theta(pn) \cdot \log M \quad \textit{with} \quad M = \max\{B_{i,j}; \quad i \in [\![1,p]\!], j \in [\![1,n]\!]\},$$

a basis $\hat{B}$ with the following characteristics:

(i) *the $\hat{B}$ basis is obtained in polynomial time in the size $\tau(B)$ of the B matrix,*
(ii) *the orthogonality defect $\rho(\hat{B})$ and the length defects $\theta_i(\hat{B})$ of the basis $\hat{B}$ satisfy*

$$\rho(\hat{B}) \leq s^{p(p-1)/2} \quad \theta_i(\hat{B}) \leq s^{p-1}.$$

The main results about the computational complexity of the LLL algorithm are the following.

Theorem II-C.2. *(Lenstra-Lenstra-Lovàsz, 1982). Let $\mathcal{L} \subset \mathbb{Z}^n$ a network given by a basis $b_1, b_2, \cdots, b_p$, and let $B := max\left\{|b_i|^2, \quad i \in [\![1,p]\!]\right\}$. Then, the number of arithmetic operations performed by the algorithm $LLL(t)$ is in $O\left(p^3 n \log_t B\right)$, and the operands are integers whose binary length is in $O(p \log B)$.*

Algorithm 1 The LLL algorithm

```
Require: A lattice $L$ given by a basis of vectors $b_1, \dots, p$.
Ensure: A Lovász-reduced basis $b$ of the lattice $L$.
  Compute the system $\hat{b}$ and the matrix $\mathcal{P}$
  Set $i := 1$;
  while $i < n$ do
    $b_{i+1} := b_{i+1} - \lceil m_{i+1,i} \rfloor b_i$
    if Lovász condition (1) is true then
      set $i := i + 1$;
    else if Lovász condition (1) is false then
      swap $b_i$ and $b_{i+1}$
      update $\hat{b}$ and $m$
      if $i \neq 1$ then $i := i - 1$
    end if
  end while
```

Finally, the following theorem proves that the LLL algorithm naturally provides an approximation algorithm for SVP, with an exponential approximation factor in the dimension.

Theorem II-C.3. *(Approximation algorithm for the Shortest Vector Problem). In a network of dimension $p \geq 2$, the algorithm $LLL(t)$ is a polynomial approximation algorithm for computing the shortest vector, with an approximation factor in s^{p-1}, where s and t are related by the relation $s^2 = 4t^2/(4 - t^2)$.*

Given these results, we turn to the question of devising a more efficient algorithm, with better scalability than the LLL algorithm.

III. Faster than LLL : an ADMM Algorithm

In this section, we take a different route in order to make the approach more scalable, namely the method described in [5]. Let us restate the problem we want to address: finding the shortest vector in a lattice corresponds to the optimisation problem

$$\min_{x \in \mathbb{Z}^J, z \in \mathbb{R}^d} \|z\|_2^2 \tag{2}$$

subject to

$$z = \sum_{j=1}^{J} a_j x_j = Ax \tag{3}$$

where $A \in \mathbb{R}^{d \times J}$ is the matrix whose columns are $a_1, \dots, a_J$.

Following the approach in [5], we introduce an ADMM type scheme that scales reasonably with the problem's dimensions.

1) An ADMM approach: The ADMM has a long history in optimisation theory [1], and was extensively used for convex minimisation problems mainly. It was revived recently, triggered by the recent needs for efficient methods for penalised regression and classification in machine learning [2].

Let us start from the following problem formulation:

$$\begin{array}{ll} \text{minimize}_{\,x \in \mathcal{X},\ z \in \mathcal{Z}} & f(x) + g(z) \\ \text{subject to} & Lx + Mz = c \end{array}$$

with variables $x \in \mathbf{R}^n$ and $z \in \mathbf{R}^m$, where $L \in \mathbf{R}^{p \times n}, M \in \mathbf{R}^{p \times m}$, and $c \in \mathbf{R}^p$. Denote

$$p^\star = \inf\{f(x) + g(z) \mid Lx + Mz = c\}.$$

Let us form the augmented Lagrangian

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Lx + Mz - c) + (\rho/2)\ \|Lx + Mz - c\|_2^2.$$

The ADMM iterations are given by

$$\begin{aligned} x^{k+1} &:= \underset{x \in \mathcal{X}}{\operatorname{argmin}}\ L_\rho\left(x, z^k, y^k\right) \\ z^{k+1} &:= \underset{z \in \mathcal{Z}}{\operatorname{argmin}}\ L_\rho\left(x^{k+1}, z, y^k\right) \\ y^{k+1} &:= y^k + \rho\left(Lx^{k+1} + Mz^{k+1} - c\right), \end{aligned}$$

The convergence proofs are available from various sources in the convex setting, i.e. when f and g are convex functions and $\mathcal{X}$ and $\mathcal{Z}$ are convex sets.

2) Application to the Shortest Vector Problem: In order to apply the ADMM approach, we need to reformulate our problem as follows.

$$\min_{x\in\mathbb{Z}^J, z\in\mathbb{R}^d} \frac{1}{2}\begin{bmatrix}x\\z\end{bmatrix}^\top \begin{bmatrix}0 & 0\\0 & I_{d\times d}\end{bmatrix}\begin{bmatrix}x\\z\end{bmatrix} \tag{4}$$

subject to

$$\begin{bmatrix}A & -I_{d\times d}\end{bmatrix}\begin{bmatrix}x\\z\end{bmatrix} = 0 \tag{5}$$

and $x_i \in \mathcal{X}_i = \mathbb{Z}$ and $z_i \in \mathbb{R}$.

We can rewrite our problem :

$$\min_{x\in\mathbb{R}^J, z\in\mathbb{R}^d, a\in\mathbb{R}^J, b\in\mathbb{R}^d} \frac{1}{2}\begin{bmatrix}x\\z\end{bmatrix}^\top \begin{bmatrix}0 & 0\\0 & I_{d\times d}\end{bmatrix}\begin{bmatrix}x\\z\end{bmatrix} + I_{\mathbb{Z}^J\times\mathbb{R}^d}((a,b)) \tag{6}$$

subject to

$$\begin{bmatrix}A & -I_{d\times d}\\ I_{J\times J} & 0_{J\times d}\\ 0_{d\times J} & I_{d\times d}\end{bmatrix}\begin{bmatrix}x\\z\end{bmatrix} - \begin{bmatrix}0_{d\times J} & 0_{d\times d}\\ I_{J\times J} & 0_{J\times d}\\ 0_{d\times J} & I_{d\times d}\end{bmatrix}\begin{bmatrix}a\\b\end{bmatrix} = \begin{bmatrix}0_{d\times 1}\\0_{J\times 1}\\0_{d\times 1}\end{bmatrix} \tag{7}$$

where the last two sets of constraints reflect the fact that we need to enforce $a = x$ and $b = z$. Here the I_S denotes the indicator function of the set S, i.e.

$$I_S(y) = \begin{cases}+\infty \text{ if } y \notin S\\ \\ 0 \text{ otherwise.}\end{cases} \tag{8}$$

The idea is then to solve

$$X = \begin{bmatrix}x\\z\end{bmatrix} \in \mathbb{R}^{J+d}. \tag{9}$$

The iterations of the ADMM algorithm of [5] are given by the three steps :

$$X^{k+\frac{1}{2}} = \text{argmin}_X \left(\frac{1}{2}X^T\begin{bmatrix}0_{J\times J} & 0_{J\times d}\\ 0_{d\times J} & I_{d\times d}\end{bmatrix}X + \frac{\rho}{2}\left\|\begin{bmatrix}A & -I_{d\times d}\\ I_{J\times J} & 0_{J\times d}\\ 0_{d\times J} & I_{d\times d}\end{bmatrix}X - \begin{bmatrix}0_{d\times J} & 0_{d\times d}\\ I_{J\times J} & 0_{J\times d}\\ 0_{d\times J} & I_{d\times d}\end{bmatrix}X^k - \begin{bmatrix}0_{d\times 1}\\0_{J\times 1}\\0_{d\times 1}\end{bmatrix} + u^k\right\|_2^2\right) \tag{10}$$

$$X^{k+1} = \mathcal{P}\left(X^{k+\frac{1}{2}} + \begin{bmatrix}0_{(J+d)\times d} & I_{(J+d)\times(J+d)}\end{bmatrix}u^k\right) \tag{11}$$

$$u^{k+1} = u^k + \begin{bmatrix}A & -I_{d\times d}\\ I_{J\times J} & 0_{J\times d}\\ 0_{d\times J} & I_{d\times d}\end{bmatrix}X^{k+\frac{1}{2}} - \begin{bmatrix}0_{d\times J} & -I_{d\times d}\\ I_{J\times J} & 0_{J\times d}\\ 0_{d\times J} & I_{d\times d}\end{bmatrix}X^k \tag{12}$$

where $\mathcal{P}_{\mathcal{X}}$ denotes the projection onto $\mathcal{X}$, $u \in \mathbb{R}^{2d+J}$ and $\rho \in \mathbb{R}$ is a scalar parameter. If $\mathcal{X}_i$ is the set of integers, $\mathcal{P}_{\mathcal{X}_i}$ rounds its argument to the nearest integer.

Equation (10) be rewritten as

$$X^{k+\frac{1}{2}} = \text{argmin}_X \left(\frac{1}{2}X^\top(M_1 + \frac{\rho}{2}M_2^\top M_2)X - \rho {Y^k}^\top M_2 X\right) \tag{13}$$

where

$$M_1 = \begin{bmatrix}0_{J\times J} & 0_{J\times d}\\ 0_{d\times J} & I_{d\times d}\end{bmatrix} \tag{14}$$

$$M_2 = \begin{bmatrix} A & -I_{d\times d} \\ I_{J\times J} & 0_{J\times d} \\ 0_{d\times J} & I_{d\times d} \end{bmatrix} \tag{15}$$

and

$$Y^k = \begin{bmatrix} 0_{d\times J} & 0_{d\times d} \\ I_{J\times J} & 0_{J\times d} \\ 0_{d\times J} & I_{d\times d} \end{bmatrix} X^k + \begin{bmatrix} 0_{d\times 1} \\ 0_{J\times 1} \\ 0_{d\times 1} \end{bmatrix} - u^k. \tag{16}$$

The derivative of

$$\frac{1}{2} X^\top (M_1 + \frac{\rho}{2} M_2^\top M_2) X - \rho {Y^k}^\top M_2 X \tag{17}$$

is

$$(M_1 + \frac{\rho}{2} M_2^\top M_2) X - \rho M_2^\top Y^k \tag{18}$$

and $X^{k+\frac{1}{2}}$ is the solution to setting this last expression to 0, i.e.

$$(M_1 + \frac{\rho}{2} M_2^\top M_2) X^{k+\frac{1}{2}} - \rho M_2^\top Y^k = 0 \tag{19}$$

which gives

$$X^{k+\frac{1}{2}} = \left(M_1 + \frac{\rho}{2} M_2^\top M_2\right)^{-1} \rho M_2^\top Y^k. \tag{20}$$

These iterations are easy to compute, except for this last equation.

A. A stochastic version of the ADMM algorithm

In order to make the algorithm more scalable, we propose to modify the iteration (21) as follows. Notice that

$$M_2^\top M_2 = \frac{\rho}{2} \begin{bmatrix} A^\top A + I_{J\times J} & -A \\ -A^\top & 0 \end{bmatrix}$$

which gives

$$\begin{aligned} M_1 + \frac{\rho}{2} M_2^\top M_2 &= \begin{bmatrix} \frac{\rho}{2} I_{J\times J} & 0 \\ 0 & I_{d\times d} \end{bmatrix} + \frac{\rho}{2} \begin{bmatrix} A^\top A & -A \\ -A^\top & 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{\rho}{2} I_{J\times J} & 0 \\ 0 & (1-\frac{\rho}{2}) I_{d\times d} \end{bmatrix} + \frac{\rho}{2} \begin{bmatrix} A^\top \\ -I_{d\times d} \end{bmatrix} \begin{bmatrix} A & -I_{d\times d} \end{bmatrix} \end{aligned}$$

Thus,

$$\begin{bmatrix} A^\top \\ -I_{d\times d} \end{bmatrix} \begin{bmatrix} A & -I_{d\times d} \end{bmatrix} = \frac{1}{d} \sum_{j=1}^{d} d \begin{bmatrix} a_j \\ e_j \end{bmatrix} \begin{bmatrix} a_j \\ e_j \end{bmatrix}^\top.$$

One simple way of proceeding in order to design a scalable approach is to replace $(M_1 + \frac{\rho}{2} M_2^\top M_2)^{-1}$ in (21) with $d\ \tilde{a}_{j_k} \tilde{a}_{j_k}^\top$, where j_k is a random integer drawn from the uniform distribution on $\{1, \ldots, d\}$, i.e.

$$X^{k+\frac{1}{2}} = \left(\begin{bmatrix} \frac{\rho}{2} I_{J\times J} & 0 \\ 0 & (1-\frac{\rho}{2}) I_{d\times d} \end{bmatrix} + \frac{d\rho}{2} \begin{bmatrix} a_{j_k} \\ e_{j_k} \end{bmatrix} \begin{bmatrix} a_{j_k} \\ e_{j_k} \end{bmatrix}^\top \right)^{-1} \rho M_2^\top Y^k, \tag{21}$$

which by the Sherman Morisson Woodbury formula gives the modified formula

$$\begin{aligned} X^{k+\frac{1}{2}} &= \begin{bmatrix} \frac{2}{\rho} I_{J\times J} & 0 \\ 0 & \frac{1}{1-\frac{\rho}{2}} I_{d\times d} \end{bmatrix} - d\ \frac{\begin{bmatrix} \frac{2}{\rho} I_{J\times J} & 0 \\ 0 & \frac{1}{1-\frac{\rho}{2}} I_{d\times d} \end{bmatrix} \begin{bmatrix} a_{j_k} \\ e_{j_k} \end{bmatrix} \begin{bmatrix} a_{j_k} \\ e_{j_k} \end{bmatrix}^\top \begin{bmatrix} \frac{2}{\rho} I_{J\times J} & 0 \\ 0 & \frac{1}{1-\frac{\rho}{2}} I_{d\times d} \end{bmatrix}}{1 + d\ \tilde{a}_{j_k}^\top \begin{bmatrix} \frac{2}{\rho} I_{J\times J} & 0 \\ 0 & \frac{1}{1-\frac{\rho}{2}} I_{d\times d} \end{bmatrix} \tilde{a}_{j_k}} \\ &= \begin{bmatrix} \frac{2}{\rho} I_{J\times J} & 0 \\ 0 & \frac{1}{1-\frac{\rho}{2}} I_{d\times d} \end{bmatrix} - d\ \frac{\begin{bmatrix} \frac{2}{\rho} a_{j_k} \\ \frac{1}{1-\frac{\rho}{2}} e_{j_k} \end{bmatrix} \begin{bmatrix} \frac{2}{\rho} a_{j_k} \\ \frac{1}{1-\frac{\rho}{2}} e_{j_k} \end{bmatrix}^\top}{1 + \frac{2d}{\rho} \|a_{j_k}\|_2^2 + \frac{d}{1-\frac{\rho}{2}}} \end{aligned}$$

The performance gain of this stochastic aglorithm in terms of scalability is studied in the next section.

IV. Simulation results

The proposed algorithm iterations have been developed using Python, a general-purpose programming language that possesses several powerful modules in this research field, which will oblige the enhancement of simplicity and scalability in future research.

Comparisons between the LLL algorithm and the ADMM algorithm have been carried out in terms of the norm value and the execution time. However, the norm value has been calculated for the last (d) values in the result vector of the ADMM algorithm, while selecting the minimum norm value of the result lattice vectors of the LLL algorithm. Moreover, execution time has been measured by calculating the average time of running both algorithms 100 times per parameter value shown in the tables (I, II, III and IV).

The detailed simulation results presented in the table demonstrate the formidable gain in efficiency provided by our ADMM based approach in various settings, both by computational time and by the optimal value achieved by the method.

References

[1] Ernesto G Birgin and José Mario Martínez. *Practical augmented Lagrangian methods for constrained optimization*. SIAM, 2014. 4

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011. 4

[3] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261(ARTICLE):515–534, 1982. 1

[4] Phong Q Nguyen and Brigitte Vallée. *The LLL algorithm*. Springer, 2010. 1

[5] Reza Takapoui, Nicholas Moehle, Stephen Boyd, and Alberto Bemporad. A simple effective heuristic for embedded mixed-integer quadratic programming. *International journal of control*, 93(1):2–12, 2020. 4, 5

d	10	12	14	16	18	20
J	10	12	14	16	18	20
K	50	50	50	50	50	50
m	-100	-100	-100	-100	-100	-100
n	100	100	100	100	100	100
ρ	0.9	0.9	0.9	0.9	0.9	0.9
ADMM smallest norm	2.789134	3.182041	4.411478	5.389277	7.09758662	7.9770541
LLL smallest norm	86.94121	115.7237	138.6946	155.179	171.4192	192.3313
ADMM time (s)	0.026558	0.034999	0.046795	0.056732	0.063938	0.078746
LLL time (s)	4.270414	10.95114	20.75395	46.5905	91.4911	185.4416

TABLE I: Parameters values and results when $d = J$

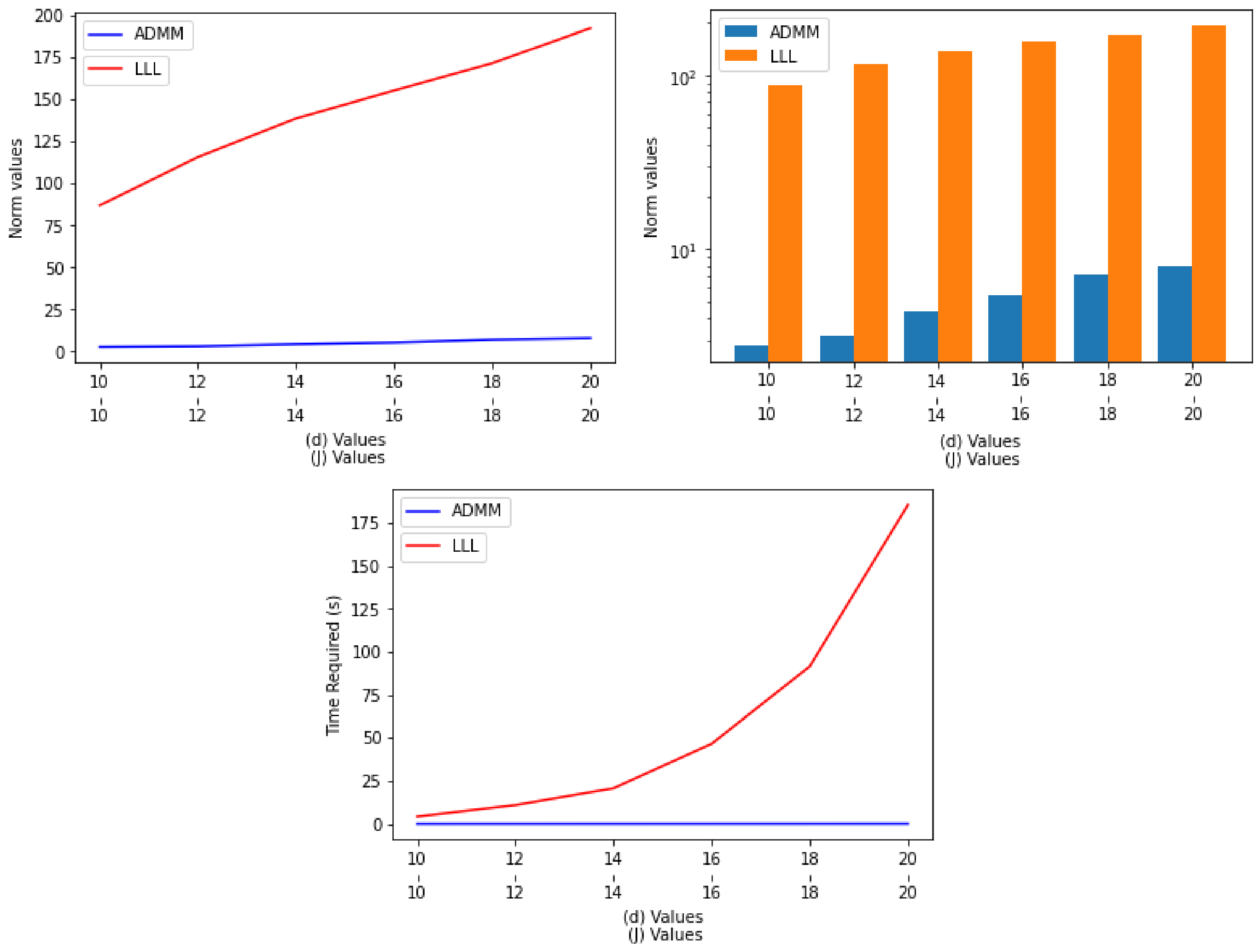

Fig. 2: ADMM and LLL Norm values and excution time of table I when $d = J$

Fig. 3

d	9	12	15	18	21	24
J	12	16	20	24	28	32
K	50	50	50	50	50	50
m	-100	-100	-100	-100	-100	-100
n	100	100	100	100	100	100
ρ	0.9	0.9	0.9	0.9	0.9	0.9
ADMM smallest norm	1.393	1.459219	1.466071	1.868535	2.049532	2.169259
LLL smallest norm	146.2937	177.2399	211.0767	233.5362	256.3163	274.5551
ADMM time (s)	0.03822	0.047063	0.056527	0.060056	0.069806	0.078271
LLL time (s)	1.081036	4.57304	16.18809	38.37236	66.50032	129.3791

TABLE II: Parameters values and results when $d < J(25\%)$

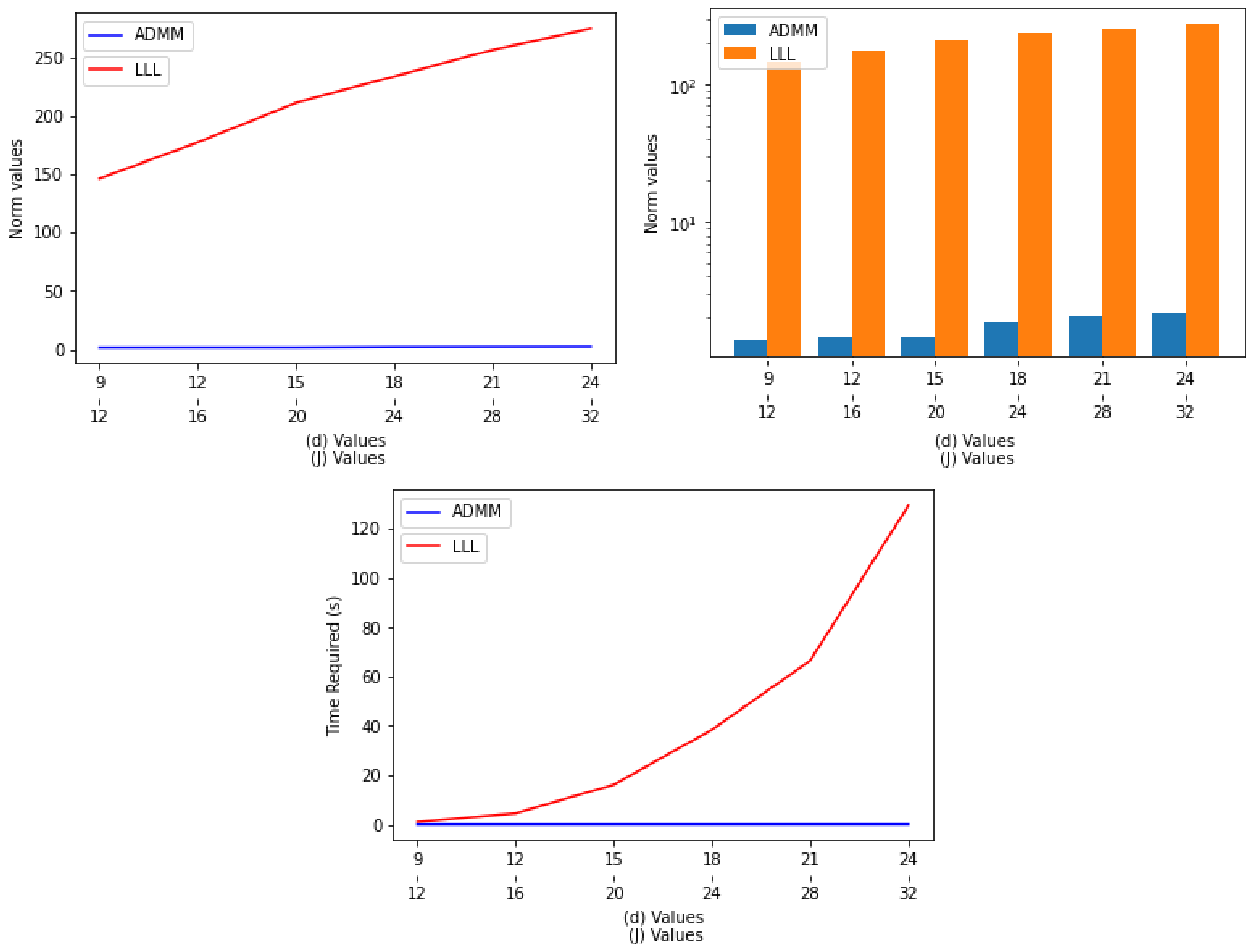

Fig. 4: ADMM and LLL Norm values and excution time of table II when $d < J(25\%)$

Fig. 5

d	5	6	7	8	9	10
J	10	12	14	16	18	20
K	50	50	50	50	50	50
m	-100	-100	-100	-100	-100	-100
n	100	100	100	100	100	100
ρ	0.9	0.9	0.9	0.9	0.9	0.9
ADMM smallest norm	8.54E-01	8.62E-01	8.81E-01	9.15E-01	0.929434	1.07E+00
LLL smallest norm	136.1071	152.1248	166.3981	185.3865	193.0321	207.7072
ADMM time (s)	0.030147	0.039204	0.04389	0.047251	0.049471	0.050611
LLL time (s)	0.074643	0.183379	0.359848	0.544247	0.850617	1.293302

TABLE III: Parameters values and results when $d < J(50\%)$

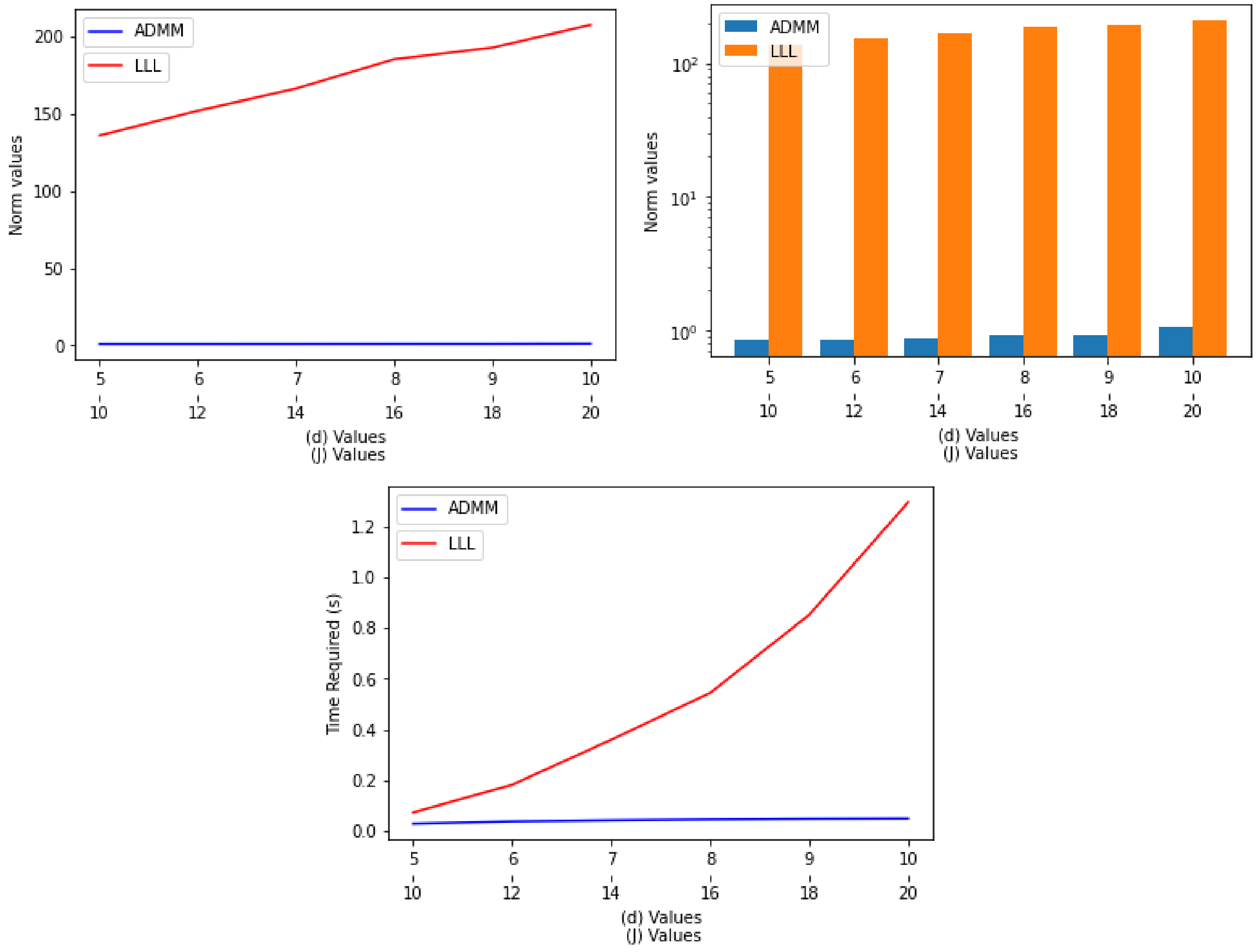

Fig. 6: ADMM and LLL Norm values and excution time of table III when $d < J(50\%)$

Fig. 7

d	3	4	5	6	7	8
J	12	16	20	24	28	32
K	50	50	50	50	50	50
m	-100	-100	-100	-100	-100	-100
n	100	100	100	100	100	100
ρ	0.9	0.9	0.9	0.9	0.9	0.9
ADMM smallest norm	3.98E-01	4.40E-01	0.410317	4.41E-01	5.22E-01	0.563505
LLL smallest norm	167.0335	196.3972	224.7434	253.5314	276.5919	284.3901
ADMM time (s)	0.026392	0.035404	0.040186	0.050001	0.055244	0.057821
LLL time (s)	0.008421	0.019914	0.053227	0.106819	0.215309	0.344097

TABLE IV: Parameters values and results when $d < J(75\%)$

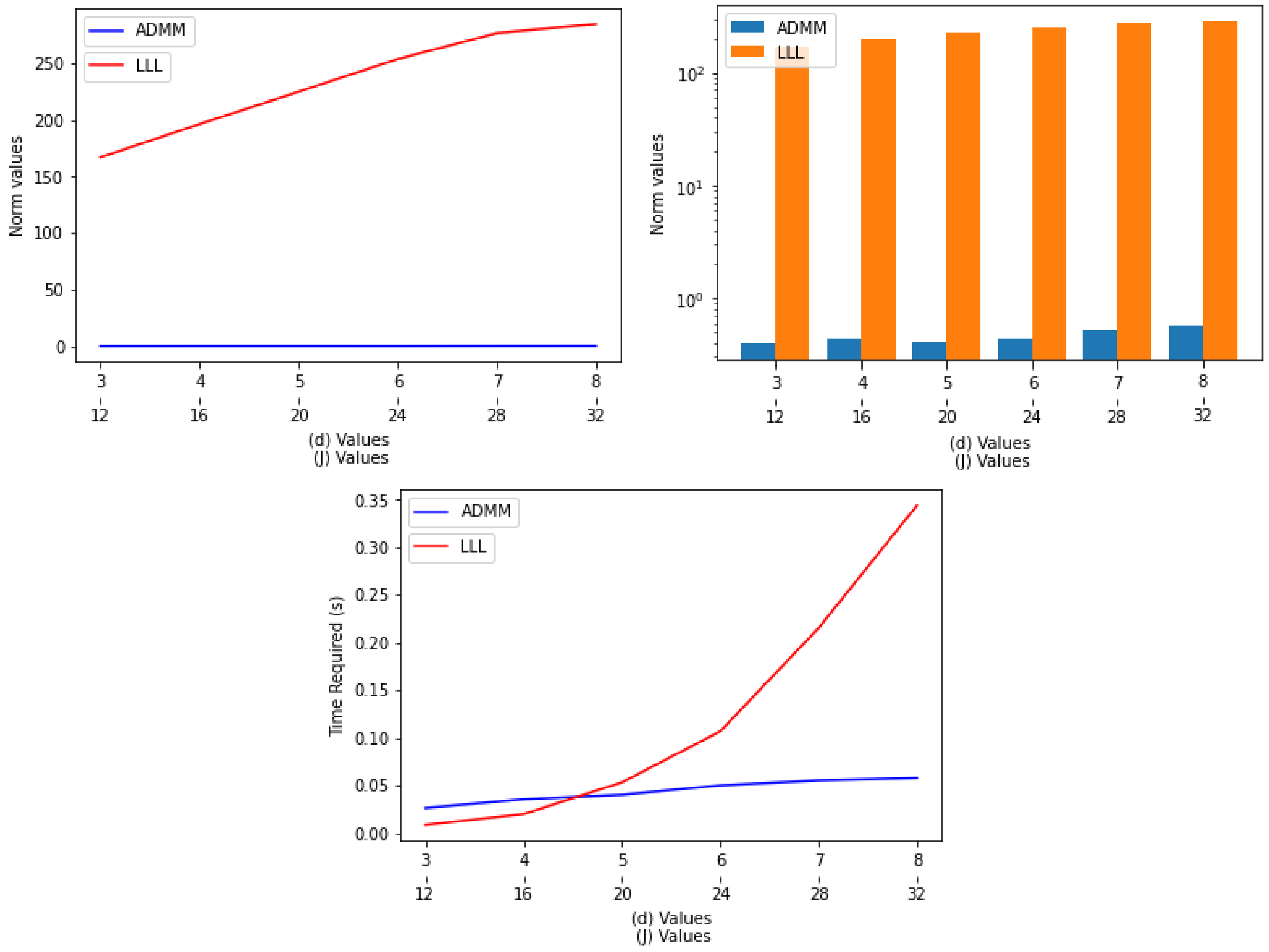

Fig. 8: ADMM and LLL Norm values and excution time of table IV when $d < J(75\%)$

Fig. 9