MILES:
MATLAB package for solving Mixed Integer LEast Squares problems

Theory and Algorithms

Xiao-Wen Chang and Tianyang Zhou

Scientific Computing Laboratory

School of Computer Science
McGill University

October 2006

1 Introduction

Let the sets of all real and integer $m \times n$ matrices be denoted by $\mathbb{R}^{m\times n}$ and $\mathbb{Z}^{m\times n}$, respectively, and the sets of real and integer n-vectors by $\mathbb{R}^n$ and $\mathbb{Z}^n$, respectively.

Given $\boldsymbol{A} \in \mathbb{R}^{m\times k}$, $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{y} \in \mathbb{R}^m$. Suppose that $[\boldsymbol{A}, \boldsymbol{B}]$ has full column rank. This MATLAB package provides a function to produce p optimal solutions to the mixed integer least squares (MILS) problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^k, \boldsymbol{z}\in\mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} - \boldsymbol{B}\boldsymbol{z}\|_2^2, \tag{1}$$

in the sense that a pair $\{\boldsymbol{x}^{(j)}, \boldsymbol{z}^{(j)}\} \in \mathbb{R}^k \times \mathbb{Z}^n$ is the j-th optimal solution if its corresponding residual norm $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{(j)} - \boldsymbol{B}\boldsymbol{z}^{(j)}\|_2$ is the j-th smallest (some of these p residual norms can be equal), thus

$$\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{(1)} - \boldsymbol{B}\boldsymbol{z}^{(1)}\|_2 \le \cdots \le \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{(j)} - \boldsymbol{B}\boldsymbol{z}^{(j)}\|_2 \le \cdots \le \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}^{(p)} - \boldsymbol{B}\boldsymbol{z}^{(p)}\|_2.$$

Here p is a parameter to be provided by a user and its default value is 1.

If the matrix $\boldsymbol{A}$ is nonexistent, (1) becomes an ordinary integer least squares (ILS) problem:

$$\min_{\boldsymbol{z}\in\mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{z}\|_2^2. \tag{2}$$

This package also provides a function to produce p optimal solutions to (2).

The purpose of this document is to present some theory and algorithms implemented by this package to solve the MILS problem (1) and the ILS problem (2). For using this package, see a separate document "MILES: MATLAB package for solving Mixed Integer LEast Squares problems, Users' Guide".

The rest of this document is organized as follows. In Section 2 we apply orthogonal transformations to transform (1) to two subproblems: an ordinary ILS problem and a real upper triangular linear system of equations (whose right hand side depends on the solution of the former). In Section 3 we give algorithms to solve the ordinary ILS problem (2). Specifically, a reduction algorithm and a search algorithm are presented. In Section 4 we give a description of the entire algorithm to solve the MILS problem (1).

We now describe other notation to be used in this document. Bold upper case letters are used to denote matrices and bold lower case letters are used to denote vectors. The identity matrix is denoted by $\boldsymbol{I}$ and its ith column is denoted by $\boldsymbol{e}_i$. MATLAB notation is used to denote a submatrix. Specifically, if $\boldsymbol{A} = (a_{ij}) \in \mathbb{R}^{m\times n}$, then $\boldsymbol{A}(i,:)$ denotes the ith row, $\boldsymbol{A}(:,j)$ the jth column, and $\boldsymbol{A}(i_1\!:\!i_2, j_1\!:\!j_2)$ the submatrix formed by rows i_1 to i_2 and columns j_1 to j_2. $\boldsymbol{D} = \mathrm{diag}(d_1, \ldots, d_n)$ specifies a diagonal matrix. For the (i,j) element of $\boldsymbol{A}$, sometimes we use a_{ij} and sometimes we use $\boldsymbol{A}(i,j)$. We use $\det(\boldsymbol{A})$ to denote the determinant of $\boldsymbol{A}$. For a scalar $z \in \mathbb{R}$, we use $\lfloor z \rceil$ to denote its nearest integer, and if there is a tie, it denotes the one with smaller magnitude. The operation $\mathrm{sgn}(z)$ returns -1 if $z \le 0$ and 1 if $z > 0$.

2 Orthogonal Transformations

To solve the MILS problem (1), we will transform it to an ILS problem and a real upper triangular linear system of equations. By solving these two sub-problems sequentially, we will obtain the MILS solution.

Suppose $\boldsymbol{A}$ has the QR factorization

$$\boldsymbol{A} = [\boldsymbol{Q}_A, \bar{\boldsymbol{Q}}_A] \begin{bmatrix} \boldsymbol{R}_A \\ \boldsymbol{0} \end{bmatrix},$$

where $[\underset{k}{\boldsymbol{Q}_A}, \underset{m-k}{\bar{\boldsymbol{Q}}_A}] \in \mathbb{R}^{m\times m}$ is orthogonal and $\boldsymbol{R}_A \in \mathbb{R}^{k\times k}$ is nonsingular upper triangular. This factorization can be computed by Householder transformations, see, e.g., [2, Chap 1] and [6, Chap 5]. Then we have

$$\begin{aligned}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}-\boldsymbol{B}\boldsymbol{z}\|_2^2 &= \left\| \begin{bmatrix}\boldsymbol{Q}_A^T\\ \bar{\boldsymbol{Q}}_A^T\end{bmatrix}\boldsymbol{y} - \begin{bmatrix}\boldsymbol{R}_A\\ \boldsymbol{0}\end{bmatrix}\boldsymbol{x} - \begin{bmatrix}\boldsymbol{Q}_A^T\boldsymbol{B}\\ \bar{\boldsymbol{Q}}_A^T\boldsymbol{B}\end{bmatrix}\boldsymbol{z}\right\|_2^2 \\ &= \|\boldsymbol{Q}_A^T\boldsymbol{y}-\boldsymbol{Q}_A^T\boldsymbol{B}\boldsymbol{z}-\boldsymbol{R}_A\boldsymbol{x}\|_2^2 + \|\bar{\boldsymbol{Q}}_A^T\boldsymbol{y}-\bar{\boldsymbol{Q}}_A^T\boldsymbol{B}\boldsymbol{z}\|_2^2. \end{aligned} \tag{3}$$

Notice that for any fixed $\boldsymbol{z}$, we can choose $\boldsymbol{x} \in \mathbb{R}^k$ such that the first term on the right hand side of (3) is equal to zero. Therefore, to solve the MILS problem (1), we first solve the ordinary ILS problem

$$\min_{\boldsymbol{z}\in\mathbb{Z}^n} \|\bar{\boldsymbol{Q}}_A^T\boldsymbol{y}-\bar{\boldsymbol{Q}}_A^T\boldsymbol{B}\boldsymbol{z}\|_2^2 \tag{4}$$

to obtain the solution $\hat{\boldsymbol{z}} \in \mathbb{Z}^n$, and then solve the upper triangular system

$$\boldsymbol{R}_A\boldsymbol{x} = \boldsymbol{Q}_A^T\boldsymbol{y} - \boldsymbol{Q}_A^T\boldsymbol{B}\hat{\boldsymbol{z}} \tag{5}$$

to obtain the real solution $\hat{\boldsymbol{x}} \in \mathbb{R}^k$. If we find p optimal integer solutions to (4), then we can obtain the corresponding p real solutions by solving (5). Thus the key is to solve the ILS problem (4). In the next section, we will present a method to solve the general ILS problem (2).

3 Solving ILS problems

In this section, we will show how to solve the ILS problem (2). For clarity, we rewrite it here:

$$\min_{\boldsymbol{z}\in\mathbb{Z}^n} \|\boldsymbol{y}-\boldsymbol{B}\boldsymbol{z}\|_2^2, \tag{6}$$

where $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ has full column rank. The entire algorithm to solve (6) consists of two processes: reduction and search. The purpose of the reduction process is to make the search process easier and more efficient. In this package, the LLL reduction [7] is used in the reduction process. Our algorithm, which is based on the modified LLL algorithms proposed in [10], is faster than the algorithms given in [4, Chap 2] and more numerically stable than the algorithm presented in [3]. The search algorithm used in this package is based on the one presented in [3], which is a modification of the Schnorr-Euchner enumeration strategy [8] based algorithm presented in [1] and is faster than the search algorithm implemented by the LAMBDA package [5]. For different reduction strategies, search algorithms and their comparisons, see [1].

3.1 Reduction

The reduction process transforms the given ILS problem (6) into a new ILS problem and its essential part is the LLL reduction which transforms $\boldsymbol{B}$ into an upper triangular matrix. The LLL reduction can be casted as a QRZ factorization (see [10]):

$$\boldsymbol{Q}^T\boldsymbol{B}\boldsymbol{Z} = \begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}, \quad \text{or} \quad \boldsymbol{B} = \boldsymbol{Q}\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}\boldsymbol{Z}^{-1} = \boldsymbol{Q}_1\boldsymbol{R}\boldsymbol{Z}^{-1}, \tag{7}$$

where $\boldsymbol{Q} = [\underset{n}{\boldsymbol{Q}_1}, \underset{m-n}{\boldsymbol{Q}_2}] \in \mathbb{R}^{m\times m}$ is orthogonal, $\boldsymbol{Z} \in \mathbb{Z}^{n\times n}$ is unimodual (i.e., $\boldsymbol{Z}$ is an integer matrix and $|\det(\boldsymbol{Z})| = 1$, thus $\boldsymbol{Z}^{-1}$ is also an integer matrix), and $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ is nonsingular upper triangular and satisfies the following two LLL reduction criteria:

$$|r_{ij}| \le \frac{1}{2}|r_{ii}|, \quad r_{ii}^2 \le r_{i,i+1}^2 + r_{i+1,i+1}^2, \quad i = 1:n-1, \quad j = i+1:n. \tag{8}$$

Here the second criterion is a special case of the more general criterion $\delta r_{ii}^2 \le r_{i,i+1}^2 + r_{i+1,i+1}^2$, where $1/4 < \delta \le 1$. For the QRZ factorization (7) where $\boldsymbol{R}$ satisfies (8), we refer to it as the LLL-QRZ factorization.

With the QRZ factorization (7), we have

$$\|\boldsymbol{y} - \boldsymbol{B}\boldsymbol{z}\|_2^2 = \|\boldsymbol{Q}_1^T\boldsymbol{y} - \boldsymbol{R}\boldsymbol{Z}^{-1}\boldsymbol{z}\|_2^2 + \|\boldsymbol{Q}_2^T\boldsymbol{y}\|_2^2.$$

Then with $\bar{\boldsymbol{y}} \triangleq \boldsymbol{Q}_1^T\boldsymbol{y}$ and $\bar{\boldsymbol{z}} \triangleq \boldsymbol{Z}^{-1}\boldsymbol{z}$, we see that (6) is equivalent to

$$\min_{\bar{\boldsymbol{z}}\in\mathbb{Z}^n} \|\bar{\boldsymbol{y}} - \boldsymbol{R}\bar{\boldsymbol{z}}\|_2^2. \tag{9}$$

Note that if $\bar{\boldsymbol{z}}$ is a solution to (9), then $\boldsymbol{z} = \boldsymbol{Z}\bar{\boldsymbol{z}}$ is a solution to (6).

In this section we will present an algorithm to compute the reduction from (6) to (9). Its main part is the computation of the LLL-QRZ factorization (7). For efficiency, we do not form the Q-factor of the LLL-QRZ factorization. When we apply an orthogonal transformation to $\boldsymbol{B}$, we also apply it to $\boldsymbol{y}$ simultaneously. In the following, we will first introduce some main components of the algorithm, and then present the entire algorithm.

3.1.1 QR factorization with minimum-column pivoting

In the reduction process, we compute the QR factorization of some matrices. In the computation, we use a column pivoting strategy, to be referred to as a minimum-column pivoting strategy, which helps to meet the second LLL reduction criterion in (8).

Suppose $\boldsymbol{C} \in \mathbb{R}^{m\times n}$ is any given matrix with full column rank. We will find an orthogonal matrix $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ and a permutation matrix $\boldsymbol{P} \in \mathbb{Z}^{n\times n}$ such that

$$\boldsymbol{Q}^T\boldsymbol{C}\boldsymbol{P} = \begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{0}\end{bmatrix}, \tag{10}$$

where $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ is nonsingular upper triangular. First, we find the column in $\boldsymbol{C}$ with the smallest 2-norm and if it is not the first column, we interchange it with the first column. We then zero $\boldsymbol{C}(2:m,1)$ by a Householder transformation $\boldsymbol{I} - 2\boldsymbol{v}\boldsymbol{v}^T/\boldsymbol{v}^T\boldsymbol{v}$, so r_{11} is determined. Then we work with the submatrix $\boldsymbol{C}(2\!:\!m, 2\!:\!n)$ and repeat the above procedure, and so on. Finally $\boldsymbol{C}$ is transformed to an upper triangular matrix.

Here we describe a general step. Suppose after the first $k-1$ steps, we obtain

$$\boldsymbol{H}_{k-1}\cdots\boldsymbol{H}_1\boldsymbol{C}\boldsymbol{P}_1\cdots\boldsymbol{P}_{k-1} = \begin{bmatrix}\boldsymbol{R}_{k-1} & \boldsymbol{U}_{k-1}\\ \boldsymbol{0} & \boldsymbol{C}_k\end{bmatrix},$$

where $\boldsymbol{R}_{k-1}$ is $(k-1)\times(k-1)$ upper triangular. In step k, we first find the column in $\boldsymbol{C}_k$ which has the smallest 2-norm, and interchange it with its first column by applying a permutation matrix $\bar{\boldsymbol{P}}_k$ to $\boldsymbol{C}_k$ from right, so we have

$$\boldsymbol{H}_{k-1}\cdots\boldsymbol{H}_1\boldsymbol{C}\boldsymbol{P}_1\cdots\boldsymbol{P}_{k-1}\boldsymbol{P}_k = \begin{bmatrix}\boldsymbol{R}_{k-1} & \bar{\boldsymbol{U}}_{k-1}\\ \boldsymbol{0} & \bar{\boldsymbol{C}}_k\end{bmatrix}, \quad \boldsymbol{P}_k = \begin{bmatrix}\boldsymbol{I}_{k-1} & \boldsymbol{0}\\ \boldsymbol{0} & \bar{\boldsymbol{P}}_k\end{bmatrix}.$$

Then we use a Householder matrix $\bar{\boldsymbol{H}}_k$ to zero $\bar{\boldsymbol{C}}_k(2:m-k+1,1)$, i.e., $\bar{\boldsymbol{H}}_k\bar{\boldsymbol{C}}_k\boldsymbol{e}_1 = r_{kk}\boldsymbol{e}_1$. Thus we have

$$\boldsymbol{H}_k\boldsymbol{H}_{k-1}\cdots\boldsymbol{H}_1\boldsymbol{C}\boldsymbol{P}_1\cdots\boldsymbol{P}_{k-1}\boldsymbol{P}_k = \begin{bmatrix}\boldsymbol{R}_k & \boldsymbol{U}_k\\ \boldsymbol{0} & \bar{\boldsymbol{C}}_{k+1}\end{bmatrix}, \quad \boldsymbol{H}_k = \begin{bmatrix}\boldsymbol{I}_{k-1} & \boldsymbol{0}\\ \boldsymbol{0} & \bar{\boldsymbol{H}}_k\end{bmatrix},$$

where $\boldsymbol{R}_k$ is $k\times k$ upper triangular. Finally we obtain the QR factorization (10), where

$$\boldsymbol{Q} = \boldsymbol{H}_1\boldsymbol{H}_2\cdots\boldsymbol{H}_n, \quad \boldsymbol{P} = \boldsymbol{P}_1\boldsymbol{P}_2\cdots\boldsymbol{P}_n.$$

Now we describe the algorithm as follows.

Algorithm 3.1 (QR Factorization with Minimum-Column Pivoting) Given $\boldsymbol{C}\in\mathbb{R}^{m\times n}$ with $m\geq n$ and $\boldsymbol{F}\in\mathbb{R}^{m\times p}$. This algorithm computes the QR factorization of $\boldsymbol{C}$ in (10) (without forming the Q-factor) and computes $\boldsymbol{F} := \boldsymbol{Q}^T\boldsymbol{F}$. The permutation matrix $\boldsymbol{P}$ is encoded in an integer vector $\boldsymbol{piv}$, i.e., $\boldsymbol{P}_j$ is the identity with rows j and $\boldsymbol{piv}(j)$ interchanged.

function [$\boldsymbol{R}$, $\boldsymbol{F}$, $\boldsymbol{piv}$] = qrmcp($\boldsymbol{C}$, $\boldsymbol{F}$)
// Compute the 2-norm squared of each column of $\boldsymbol{C}$
for $j = 1{:}n$
  $s(j) = \boldsymbol{C}(1{:}m,j)^T\boldsymbol{C}(1{:}m,j)$
end
for $k = 1{:}n$
  $q = \arg\min_{k\leq j\leq n} s(j)$
  $\boldsymbol{piv}(k) = q$
  if $q > k$
    Interchange $\boldsymbol{C}(:,k)$ and $\boldsymbol{C}(:,q)$
    Interchange $s(k)$ and $s(q)$
  end
  Use $\boldsymbol{C}(k{:}m,k)$ to find the Householder vector $\boldsymbol{v}_k$
  $\boldsymbol{C}(k{:}m,k{:}n) = \boldsymbol{C}(k{:}m,k{:}n) - (2\boldsymbol{v}_k/\boldsymbol{v}_k^T\boldsymbol{v}_k) * (\boldsymbol{v}_k^T * \boldsymbol{C}(k{:}m,k{:}n))$
  $\boldsymbol{F}(k{:}m,:) = \boldsymbol{F}(k{:}m,:) - (2\boldsymbol{v}_k/\boldsymbol{v}_k^T\boldsymbol{v}_k) * (\boldsymbol{v}_k^T * \boldsymbol{F}(k{:}m,:))$
// Compute the 2-norm squared of each column of $\boldsymbol{C}(k+1{:}m,k+1{:}n)$
  for $j = k+1{:}n$
    $s(j) = s(j) - \boldsymbol{C}(k,j)^2$
  end
end
// Set $\boldsymbol{R}$ to be the upper triangular part of $\boldsymbol{C}$
$\boldsymbol{R}$ = triu($\boldsymbol{C}$)

3.1.2 Integer Gauss transformations for off-diagonal size reduction

To reduce the off-diagonal entries of the upper triangular matrix $\boldsymbol{R}$ to meet the first LLL reduction criterion in (8), we use the so-called integer Gauss transformations or integer Gauss matrices (see, e.g., [9]), which have the following form:

$$\boldsymbol{Z}_{ij} = I - \alpha\boldsymbol{e}_i\boldsymbol{e}_j^T, \quad i<j, \quad \alpha \text{ is an integer.}$$

It is easy to show that $\boldsymbol{Z}_{ij}$ is unimodual.

Applying $\boldsymbol{Z}_{ij}$ $(i < j)$ to $\boldsymbol{R}$ from the right gives

$$\bar{\boldsymbol{R}} \triangleq \boldsymbol{R}\boldsymbol{Z}_{ij} = \boldsymbol{R} - \alpha \boldsymbol{R}\boldsymbol{e}_i\boldsymbol{e}_j^T.$$

Thus $\bar{\boldsymbol{R}}$ is the same as $\boldsymbol{R}$, except that

$$\bar{r}_{kj} = r_{kj} - \alpha r_{ki}, \quad k = 1, \ldots, i.$$

Taking $\alpha = \lfloor r_{ij}/r_{ii} \rceil$ to ensure $|\bar{r}_{ij}| \leq |\bar{r}_{ii}|/2$.

Algorithm 3.2 (Integer Gauss Transformations) Given a nonsingular upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$, a matrix $\boldsymbol{Z} \in \mathbb{Z}^{m\times n}$, and two column indices j_1 and j_2 with $j_1 \leq j_2$, this algorithm applies integer Gauss transformations $\boldsymbol{Z}_{ij}$ to $\boldsymbol{R}$ such that after the transformations $|r_{ij}| \leq |r_{ii}|/2$ for $j = j_1\!:\!j_2$, $i = j-1\!:\!-1\!:\!1$ and it also computes $\boldsymbol{Z} := \boldsymbol{Z}\cdot\Pi_{j=j_1}^{j_2}\Pi_{i=j-1}^{1}\boldsymbol{Z}_{ij}$.

function: $[\boldsymbol{R}, \boldsymbol{Z}] = \text{gauss}(\boldsymbol{R}, \boldsymbol{Z}, j_1, j_2)$
for $j = j_1 : j_2$
  for $i = j-1\!:\!-1\!:\!1$
    $\alpha = \lfloor \boldsymbol{R}(i,j)/\boldsymbol{R}(i,i) \rceil$
    if $\alpha \neq 0$
      $\boldsymbol{R}(1\!:\!i, j) = \boldsymbol{R}(1\!:\!i, j) - \alpha \boldsymbol{R}(1\!:\!i, i)$
      $\boldsymbol{Z}(:, j) = \boldsymbol{Z}(:, j) - \alpha \boldsymbol{Z}(:, i)$
    end
  end
end

3.1.3 Column reordering

In order to meet the second LLL reduction criterion in (8), we use the minimum-column pivoting strategy to do column reordering. We first find a submatrix $\boldsymbol{R}(:, j_1 : j_2)$ which "most" violates the second LLL reduction criterion, then we reduce its off-diagoanl entries by integer Gauss transformations and apply the QR factorization with minimum-column pivoting to $\boldsymbol{R}(j_1\!:\!j_2, j_1\!:\!j_2)$. The columns $j_1 : j_2$ of $\boldsymbol{R}$ are then reordered.

Now we show how to choose j_1 and j_2 exactly. We choose the index j_2 such that $|r_{j_2-1,j_2-1}|$ will decrease most if columns $j_2 - 1$ and j_2 are interchanged, i.e., j_2 is defined by

$$j_2 = \arg\min_{2\leq j\leq n} \{|\bar{r}_{j-1,j-1}/r_{j-1,j-1}| : |\bar{r}_{j-1,j-1}/r_{j-1,j-1}| < 1\}, \tag{11}$$

where

$$|\bar{r}_{j-1,j-1}| \triangleq \sqrt{\tilde{r}_{j-1,j}^2 + r_{j,j}^2}, \quad \tilde{r}_{j-1,j} \triangleq r_{j-1,j} - \left\lfloor \frac{r_{j-1,j}}{r_{j-1,j-1}} \right\rceil r_{j-1,j-1}.$$

Note that $\tilde{r}_{j-1,j}$ is the value of the reduced $r_{j-1,j}$ by the integer Gauss transformation $\boldsymbol{Z}_{j-1,j}$ (so $|\tilde{r}_{j-1,j}| \leq |r_{j-1,j-1}|/2$) and $\bar{r}_{j-1,j-1}$ is the new value of $r_{j-1,j-1}$ if columns $j-1$ and j of $\boldsymbol{R}$ are interchanged and $\boldsymbol{R}$ is brought back to an upper triangular matrix by a Givens rotation from the left. For efficiency, we do not actually perform the integer Gauss transformation on $\boldsymbol{R}$ and the column permutation; instead, we just compute the value of $\bar{r}_{j-1,j-1}$.

After obtaining j_2, we apply integer Gauss transformations $\boldsymbol{Z}_{i,j_2}$ for $i = j_2 - 1 : -1 : 1$ to $\boldsymbol{R}$ to reduce the off-diagonal entries in column j_2. Then we find j_1 such that

$$j_1 = \min\{j : |r_{kk}| > \|\boldsymbol{R}(k\!:\!j_2, j_2)\|_2,\ k = j : j_2 - 1\}. \tag{12}$$

Note that each column j for $j = j_1 : j_2 - 1$ and column j_2 are not in correct order while column $j_1 - 1$ and column j_2 are in correct order.

3.1.4 Reduction algorithm

Now we introduce the complete process for reduction. First we compute the QR factorization of $\boldsymbol{B}$ with minimum-column pivoting by Algorithm 3.1. Then we start to work with the upper triangular matrix $\boldsymbol{R}$. We try to find the index j_2 satisfying (11). We assume that such a j_2 exists, otherwise the columns of $\boldsymbol{R}$ are already in good order, i.e., after off-diagonal reduction, $\boldsymbol{R}$ will satisfy the LLL reduction criteria (8). Then we apply integer Gauss transformations to the j_2-th column of $\boldsymbol{R}$ to reduce its off-diagonal entries and find the index j_1 satisfying (12). After that, we apply integer Gauss transformations to reduce the off-diagonal entries of $\boldsymbol{R}$ in the submatrix $\boldsymbol{R}(:, j_1{+}1 : j_2 - 1)$. Doing this is to make the later column reordering meaningful (note that in the second LLL reduction criterion in (8) the superdiagonal entries satisfy the first criterion) and to make the reduction process more numerically stable (this helps to avoid producing large numbers during the reduction process). Then we compute the QR factorization of $\boldsymbol{R}(j_1 : j_2, j_1 : j_2)$ with minimum column pivoting and reorder the columns of $\boldsymbol{R}(:, j_1 : j_2)$. After this, we apply integer Gauss transformations to reduce the off-diagonal entries in $\boldsymbol{R}(:, j_1 : j_2)$ for numerical stability. Then we continue finding a new submatrix and repeat the above process until no index j_2 satisfying (11) can be found. Finally, we apply integer Gauss transformations to all off-diagonal entries of $\boldsymbol{R}$, resulting in an upper triangular $\boldsymbol{R}$ satisfying the LLL reduction criteria (8).

Algorithm 3.3 (Reduction) Given $\boldsymbol{B} \in \mathbb{R}^{m \times n}$ with full column rank and $\boldsymbol{y} \in \mathbb{R}^m$. This algorithm computes the LLL-QRZ factorization (7) (without forming the Q-factor) and $\boldsymbol{y} := \boldsymbol{Q}^T \boldsymbol{y}$.

function: $[\boldsymbol{R}, \boldsymbol{Z}, \boldsymbol{y}] = \text{reduction}(\boldsymbol{B}, \boldsymbol{y})$
$\boldsymbol{Z} = \boldsymbol{I}_n$
// Compute the QR factorization of $\boldsymbol{B}$ with minimum-column pivoting
$[\boldsymbol{R}, \boldsymbol{y}, \boldsymbol{piv}] = \text{qrmcp}(\boldsymbol{B}, \boldsymbol{y})$
// Update $\boldsymbol{Z}$ using $\boldsymbol{piv}$
for $j = 1 : n$
    Interchange $\boldsymbol{Z}(:, j)$ with $\boldsymbol{Z}(:, \boldsymbol{piv}(j))$
end
while true
    // Find j_2
    $j_2 = 0$
    $minratio = 1$
    for $j = 2 : n$
        if $|\boldsymbol{R}(j, j)/\boldsymbol{R}(j{-}1, j{-}1)| < minratio$
            $\alpha = \lfloor \boldsymbol{R}(j{-}1, j)/\boldsymbol{R}(j{-}1, j{-}1) \rceil$
            $\eta = \boldsymbol{R}(j{-}1, j) - \alpha \boldsymbol{R}(j{-}1, j{-}1)$
            $ratio = \sqrt{\boldsymbol{R}(j, j)^2 + \eta^2}/|\boldsymbol{R}(j{-}1, j{-}1)|$
            if $ratio < minratio$
                $j_2 = j$
                $minratio = ratio$
            end
        end

end
if $j_2 = 0$ // No j_2 can be found
break the while loop
end
// Perform off-diagonal size reduction for $\boldsymbol{R}(:, j_2)$
$[\boldsymbol{R}, \boldsymbol{Z}] = \text{gauss}(\boldsymbol{R}, \boldsymbol{Z}, j_2, j_2)$
// Find j_1
$j_1 = j_2 - 1$
while $j_1 > 1$ and $|\boldsymbol{R}(j_1-1, j_1-1)| > \|\boldsymbol{R}(j_1-1:j_2, j_2)\|_2$
$j_1 = j_1 - 1$
end
// Perform off-diagonal size reduction for $\boldsymbol{R}(:, j_1+1:j_2-1)$
$[\boldsymbol{R}, \boldsymbol{Z}] = \text{gauss}(\boldsymbol{R}, \boldsymbol{Z}, j_1 + 1, j_2 - 1)$
// Apply QRMCP to $\boldsymbol{R}(j_1:j_2, j_1:j_2)$ and update $\boldsymbol{y}(j_1:j_2)$ and $\boldsymbol{R}(j_1:j_2, j_2+1:n)$
$[\boldsymbol{R}(j_1:j_2, j_1:j_2), \boldsymbol{T}, \boldsymbol{piv}] = \text{qrmcp}(\boldsymbol{R}(j_1:j_2, j_1:j_2), [\boldsymbol{y}(j_1:j_2), \boldsymbol{R}(j_1:j_2, j_2+1:n)])$
$\boldsymbol{y}(j_1:j_2) = \boldsymbol{T}(:, 1)$, $\boldsymbol{R}(j_1:j_2, j_2+1:n) = \boldsymbol{T}(:, 2:n-j_2+1)$
// Reorder columns of $\boldsymbol{R}(1:j_1-1, j_1:j_2)$ and $\boldsymbol{Z}$ using $\boldsymbol{piv}$
for $j = j_1 : j_2$
Interchange $\boldsymbol{R}(1:j_1-1, j)$ with $\boldsymbol{R}(1:j_1-1, j_1 + \boldsymbol{piv}(j - j_1 + 1)-1)$
Interchange $\boldsymbol{Z}(:, j)$ with $\boldsymbol{Z}(:, j_1 + \boldsymbol{piv}(j - j_1 + 1)-1)$
end
// Perform off-diagonal size reductions for $\boldsymbol{R}(:, j_1:j_2)$
$[\boldsymbol{R}, \boldsymbol{Z}] = \text{gauss}(\boldsymbol{R}, \boldsymbol{Z}, j_1, j_2)$
end
// Perform off-diagonal size reduction for $\boldsymbol{R}$
$[\boldsymbol{R}, \boldsymbol{Z}] = \text{gauss}(\boldsymbol{R}, \boldsymbol{Z}, 2, n)$

In order to reduce data communication time, in the implementation of the above algorithm, we perform the integer Gauss transformations directly in the above function without calling the separate function `gauss`.

3.2 Search

After the reduction, the ILS problem (6) is transformed to the new ILS problem (9). To solve (9), a search strategy is used to enumerate possible $\boldsymbol{z} \in \mathbb{Z}^n$. Our package has the option to find multiple optimal solutions to (6), which have the smallest residual norms. But we will first show how to find the optimal solution, which has the smallest residual norm, and then show how to extend the ideas to find multiple optimal solutions.

To simplify notation, we rewrite (9) as

$$\min_{\boldsymbol{z} \in \mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2^2. \tag{13}$$

Suppose that the optimal $\boldsymbol{z}$ satisfies the following bound

$$f(\boldsymbol{z}) \triangleq \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2^2 < \beta,$$

or equivalently

$$\sum_{k=1}^{n}(y_k - \sum_{j=k}^{n} r_{kj}z_j)^2 < \beta. \tag{14}$$

This is an ellipsoid and our problem is to search this ellipsoid to find the optimal solution.

Define

$$c_n \triangleq y_n/r_{nn}, \quad c_k \triangleq (y_k - \sum_{j=k+1}^{n} r_{kj}z_j)/r_{kk}, \quad k = n-1, \ldots, 1. \tag{15}$$

Notice that c_k depends on $z_n, z_{n-1}, \ldots, z_{k+1}$ and it is determined when the latter are determined. Then (14) can be rewritten as

$$\sum_{k=1}^{n} r_{kk}^2(z_k - c_k)^2 < \beta.$$

From this, it follows that

$$\text{level } n: \ r_{nn}^2(z_n - c_n)^2 < \beta, \tag{16}$$

$$\vdots$$

$$\text{level } k: \ r_{k,k}^2(z_k - c_k)^2 < \beta - \sum_{i=k+1}^{n} r_{ii}^2(z_i - c_i)^2, \tag{17}$$

$$\vdots$$

$$\text{level } 1: \ r_{11}^2(z_1 - c_1)^2 < \beta - \sum_{i=2}^{n} r_{ii}^2(z_i - c_i)^2. \tag{18}$$

Based on these bounds a search procedure can be developed.

First at level n, we choose $z_n = \lfloor c_n \rceil$. If it does not satisfy the bound (16), no any other integer will satisfy it, thus there is no any integer point within the ellipsoid. This will not happen if the initial ellipsoid bound β is large enough (see later comments). If it satisfies the bound, we proceed to level $n-1$. At this level, we compute c_{n-1} by (15) and choose $z_{n-1} = \lfloor c_{n-1} \rceil$. If z_{n-1} does not satisfy the bound (17) with $k = n-1$, then we move back to level n and choose z_n to be the second nearest integer to c_n, and so on; otherwise, we proceed to level $n-2$. We continue this procedure until we reach level 1 and obtain an integer point $\hat{\boldsymbol{z}}$. We store this point and update the bound β by taking $\beta = \|\bar{\boldsymbol{y}} - \boldsymbol{R}\hat{\boldsymbol{z}}\|_2^2$. Note that the ellipsoidal region is shrunk—this is crucial for search efficiency. Then we start to try to find an integer point within the new ellipsoid. The basic idea is to update the latest found integer point $\hat{\boldsymbol{z}}$. Obviously, we cannot update only its first entry z_1, since at level 1, we cannot find any new integer z_1 to satisfy (18), which is now an equality. Thus we move up to level 2 to update the value of z_2 by choosing z_2 to be the next nearest integer to c_2 ("next" means "next to $\hat{z}_2$"). If it satisfies the bound at level 2, we move down to level 1 to update the value of z_1 and obtain a new integer point (note that z_2 has just been updated and $z_3, \ldots, z_n$ are the same as those corresponding entries of $\hat{\boldsymbol{z}}$), otherwise we move up to level 3 to update the value of z_3, and so on. Finally, when we fail to find a new value for z_n to satisfy the bound (16) at level n, the search process stops and the latest found integer point is the optimal solution we seek.

In our algorithm, the initial bound β is set to be ∞ and we refer to the first found integer point $\hat{\boldsymbol{z}}$ as the Babai integer point. Note that $\boldsymbol{R}\hat{\boldsymbol{z}}$ is called the Babai (lattice) point in the literature, see, e.g., [1].

The above process is summarized as follows:

Algorithm 3.4 Given nonsingular upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{y} \in \mathbb{R}^n$. This algorithm finds the optimal solution $\hat{\boldsymbol{z}}$ to $\min_{\boldsymbol{z}} \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2^2$.

1. (Initialization) Set $k = n$, $\beta = \infty$.
2. Compute $\boldsymbol{c}(k) = (\boldsymbol{y}(k) - \sum_{j=k+1}^{n} \boldsymbol{R}(k,j)\boldsymbol{z}(j))/\boldsymbol{R}(k,k)$, $\boldsymbol{z}(k) = \lfloor \boldsymbol{c}(k) \rceil$, $\boldsymbol{d}(k) = \mathrm{sgn}(\boldsymbol{c}(k) - \boldsymbol{z}(k))$.
3. (Main step) If $\sum_{i=k}^{n} \boldsymbol{R}(i,i)^2(\boldsymbol{z}(i) - \boldsymbol{c}(i))^2 > \beta$, then go to Step 4. Elseif $k > 1$, then $k = k - 1$ and go to Step 2. Else ($k = 1$), go to Step 5.
4. (Outside ellipsoid) If $k = n$, terminate. Else, set $k = k + 1$ and go to Step 6.
5. (A valid point is found) Set $\beta = \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2^2$, save $\hat{\boldsymbol{z}} = \boldsymbol{z}$. Then, let $k = k + 1$ and go to Step 6.
6. (Enumeration at level k) Set $\boldsymbol{z}(k) = \boldsymbol{z}(k) + \boldsymbol{d}(k)$, $\boldsymbol{d}(k) = -\boldsymbol{d}(k) - \mathrm{sgn}(\boldsymbol{d}(k))$ and go to Step 3

Algorithm 3.4 finds only the optimal solution. We now show how to modify the above process to find p optimal ILS solutions. At the beginning we set β to be infinity. Denote the first integer point obtained by the search process (i.e., the Babai point) by $\boldsymbol{z}^{(1)}$. Then we take the second integer point $\boldsymbol{z}^{(2)}$ to be identical to $\boldsymbol{z}^{(1)}$ except that the first entry in $\boldsymbol{z}^{(2)}$ is taken as the second nearest integer to c_1 (note that the first entry in $\boldsymbol{z}^{(1)}$ is the nearest integer to c_1). The third $\boldsymbol{z}^{(3)}$ is chosen to be the same as $\boldsymbol{z}^{(1)}$ except that its first entry is taken as the third nearest integer to c_1, and so on. In this way we obtain p integer points $\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(p)}$. Obviously we have $f(\boldsymbol{z}^{(1)}) \le f(\boldsymbol{z}^{(2)}) \cdots \le f(\boldsymbol{z}^{(p)})$. Then we shrink the ellipsoidal region by setting $\beta = f(\boldsymbol{z}^{(p)})$ and start to search a new integer point within the new ellipsoid. Suppose the new integer point we have found is $\boldsymbol{z}^{(\mathrm{new})}$ and $f(\boldsymbol{z}^{(j-1)}) \le f(\boldsymbol{z}^{(\mathrm{new})}) \le f(\boldsymbol{z}^{(j)})$. We remove the point $\boldsymbol{z}^{(p)}$ and rename $\boldsymbol{z}^{(\mathrm{new})}, \boldsymbol{z}^{(j)}, \cdots, \boldsymbol{z}^{(p-1)}$ as $\boldsymbol{z}^{(j)}, \boldsymbol{z}^{(j+1)}, \cdots, \boldsymbol{z}^{(p)}$, respectively. Then we shrink the ellipsoidal region again by setting $\beta = f(\boldsymbol{z}^{(p)})$ and continue the above process until we cannot find a new integer point. Finally we end up with p optimal ILS points. Here we would like to point out that after an integer point is found, the algorithm tries to update its first entry to obtain the next integer point. This is different from the process of finding the single optimal solution, in which after an integer point is found the algorithm moves to level 2 and try to update its second entry.

We would also like to modify Algorithm 3.4 to make it more efficient by using some techniques introduced in [1], see also [3]. For computational efficiency, we do not want to recompute something which has been computed if possible. In Step 2 of Algorithm 3.4, some partial summation in $\sum_{j=k+1}^{n} \boldsymbol{R}(k,j)\boldsymbol{z}(j)$, say $\sum_{j=l}^{n} \boldsymbol{R}(k,j)\boldsymbol{z}(j)$ for $l > k+1$, may have been obtained in a previous step when the algorithm was at level k, and so it should not be recomputed in the current step at level k. What we can do is that after we have determined $\boldsymbol{z}(j)$, we compute $\boldsymbol{R}(:,j)\boldsymbol{z}(j)$ and add it to $\boldsymbol{R}(:,j+1:n)\boldsymbol{z}(j+1:n)$, which was obtained after $\boldsymbol{z}(j+1)$ was determined, to obtain $\boldsymbol{R}(:,j:n)\boldsymbol{z}(j:n)$. Since the values of all $\boldsymbol{z}(j)$ may be updated in the process of finding an integer point, we use a matrix $\boldsymbol{S}(1:n, 1:n)$ to store the above results: $\boldsymbol{S}(:,k) = \boldsymbol{R}(:,k:n)\boldsymbol{z}(k:n)$. Thus we see that $\boldsymbol{S}(k,k+1) = \sum_{j=k+1}^{n} \boldsymbol{R}(k,j)\boldsymbol{z}(j)$, which will be used in the modified algorithm. In Step 3 of Algorithm 3.4, obviously we can use the result obtained at level $k+1$ to compute the partial squared residual norm $\sum_{i=k}^{n} \boldsymbol{R}(i,i)^2(\boldsymbol{z}(i) - \boldsymbol{c}(i))^2$ at level k and keep this value for later uses.

Now the complete and more efficient search algorithm can be described as follows:

Algorithm 3.5 (Search) Given a nonsingular upper triangular $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{y} \in \mathbb{R}^n$. This algorithm searches p optimal solutions to the ILS problem $\min_{\boldsymbol{z}\in\mathbb{Z}^n} \|\boldsymbol{y} - \boldsymbol{R}\boldsymbol{z}\|_2^2$. The j-th optimal solution is given by $\hat{\boldsymbol{Z}}(:,j)$ for $j = 1:p$.

function: $\hat{\boldsymbol{Z}} = \text{search}(\boldsymbol{R}, \boldsymbol{y}, p)$
// Initialization
$\boldsymbol{z} = zeros(n, 1)$ // the current point
$\boldsymbol{c} = zeros(n, 1)$ // $\boldsymbol{c}(k) = (\boldsymbol{y}(k) - \boldsymbol{R}(k, k+1\!:\!n)\boldsymbol{z}(k+1\!:\!n))/\boldsymbol{R}(k,k)$
$\boldsymbol{d} = zeros(n, 1)$ // $\boldsymbol{d}(k)$ is used to compute $\boldsymbol{z}(k)$
$\boldsymbol{prsd} = zeros(n, 1)$ // $\boldsymbol{prsd}(k) = \sum_{i=k+1}^{n} \boldsymbol{R}(i,i)^2(\boldsymbol{z}(i) - \boldsymbol{c}(i))^2$
$\boldsymbol{S} = zeros(n, n+1)$ // $\boldsymbol{S}(:, k) = \boldsymbol{R}(:, k\!:\!n)\boldsymbol{z}(k\!:\!n)$
$\hat{\boldsymbol{Z}} = zeros(n, p)$ // p candidate solutions in order of nondecreasing residual norms
$\boldsymbol{rsd} = zeros(p, 1)$ // squared residual norms of the p candidate solutions in $\hat{\boldsymbol{Z}}$
$\beta = \infty$ // ellipsoid bound
$ncand = 0$ // number of candidate solutions
$\boldsymbol{c}(n) = \boldsymbol{y}(n)/\boldsymbol{R}(n,n)$; $\boldsymbol{z}(n) = \lfloor \boldsymbol{c}(n) \rceil$
$\gamma = \boldsymbol{R}(n,n)(\boldsymbol{c}(n) - \boldsymbol{z}(n))$; $\boldsymbol{d}(n) = \text{sgn}(\boldsymbol{c}(n) - \boldsymbol{z}(n))$
$k = n$
while *true*
    $newprsd = \boldsymbol{prsd}(k) + \gamma^2$ // compute $\sum_{i=k}^{n} \boldsymbol{R}(i,i)^2(\boldsymbol{z}(i) - \boldsymbol{c}(i))^2$
    if $newprsd < \beta$
        if $k \neq 1$ // move to level $k-1$
            $\boldsymbol{S}(1\!:\!k, k) = \boldsymbol{R}(1\!:\!k, k) * \boldsymbol{z}(k) + \boldsymbol{S}(1\!:\!k, k+1)$
            $k = k - 1$
            $\boldsymbol{prsd}(k) = newprsd$
            $\boldsymbol{c}(k) = (\boldsymbol{y}(k) - \boldsymbol{S}(k, k+1))/\boldsymbol{R}(k,k)$; $\boldsymbol{z}(k) = \lfloor \boldsymbol{c}(k) \rceil$
            $\gamma = \boldsymbol{R}(k,k)(\boldsymbol{c}(k) - \boldsymbol{z}(k))$; $\boldsymbol{d}(k) = \text{sgn}(\boldsymbol{c}(k) - \boldsymbol{z}(k))$
        else // $k = 1$, a new point is found, update the set of candidate solutions
            if $ncand < p$ // add the new point
                $ncand = ncand + 1$
                $\hat{\boldsymbol{Z}}(:, ncand) = \boldsymbol{z}$; $\boldsymbol{rsd}(ncand) = newprsd$
                if $ncand = p$, $\beta = \boldsymbol{rsd}(p)$; **end**
            else // $ncand < p$, insert the new point and remove the worst one
                $i = 1$
                while $i < p$ and $\boldsymbol{rsd}(i) \leq newprsd$; $i = i + 1$; **end**
                $\hat{\boldsymbol{Z}}(:, i\!:\!p) = [\boldsymbol{z}, \hat{\boldsymbol{Z}}(:, i\!:\!p-1)]$
                $\boldsymbol{rsd}(i\!:\!p) = [newprsd; \boldsymbol{rsd}(i\!:\!p-1)]$; $\beta = \boldsymbol{rsd}(p)$
            end
            $\boldsymbol{z}(1) = \boldsymbol{z}(1) + \boldsymbol{d}(1)$; $\gamma = \boldsymbol{R}(1,1)(\boldsymbol{c}(1) - \boldsymbol{z}(1))$; $\boldsymbol{d}(1) = -\boldsymbol{d}(1) - \text{sgn}(\boldsymbol{d}(1))$
        end
    else // $newprsd \geq \beta$
        if $k = n$ // the p optimal solutions have been found
            break the while loop
        else // $k \neq n$, move back to level $k+1$
            $k = k + 1$
            $\boldsymbol{z}(k) = \boldsymbol{z}(k) + \boldsymbol{d}(k)$; $\gamma = \boldsymbol{R}(k,k) * (\boldsymbol{c}(k) - \boldsymbol{z}(k))$; $\boldsymbol{d}(k) = -\boldsymbol{d}(k) - \text{sgn}(\boldsymbol{d}(k))$
        end
    end
end

3.3 ILS algorithm

With the reduction algorithm and the search algorithm, we can give a complete algorithm for solving the ILS problem (6).

Algorithm 3.6 (ILS Solution) Given $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ with full column rank and $\boldsymbol{y} \in \mathbb{R}^{m\times n}$. This algorithm computes the p optimal solutions to the ILS problem $\min_{\boldsymbol{z}\in\mathbb{Z}^n} \|\boldsymbol{y}-\boldsymbol{B}\boldsymbol{z}\|_2^2$. The j-th optimal solution is given by $\hat{\boldsymbol{Z}}(:,j)$ for $j = 1:p$.

function: $\hat{\boldsymbol{Z}} = \text{ils}(\boldsymbol{B}, \boldsymbol{y}, p)$
$[\boldsymbol{R},\boldsymbol{Z},\boldsymbol{y}] = \text{reduction}(\boldsymbol{B},\boldsymbol{y})$
$\hat{\boldsymbol{Z}} = \text{search}(\boldsymbol{R}, \boldsymbol{y}(1:n), p)$
$\hat{\boldsymbol{Z}} = \boldsymbol{Z} * \hat{\boldsymbol{Z}}$

4 Solving MILS problems

Based on the procedure we described in Section 2, we give a complete algorithm for solving the MILS problem (1).

Algorithm 4.1 (MILS Solution) Given $\boldsymbol{A} \in \mathbb{R}^{m\times k}$, $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{y} \in \mathbb{R}^m$. Let $[\boldsymbol{A}, \boldsymbol{B}]$ have full column rank. This algorithm computes p optimal solutions to $\min_{\boldsymbol{x}\in\mathbb{R}^k,\boldsymbol{z}\in\mathbb{Z}^n} \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}-\boldsymbol{B}\boldsymbol{z}\|_2^2$. The outputs are two matrices $\hat{\boldsymbol{X}}$ and $\hat{\boldsymbol{Z}}$, and the pair $\{\hat{\boldsymbol{X}}(:,j), \hat{\boldsymbol{Z}}(:,j)\}$ is the j-th optimal solution, for $j = 1:n$.

function: $[\hat{\boldsymbol{X}}, \hat{\boldsymbol{Z}}] = \text{mils}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{y}, p)$
Compute the QR factorization of $\boldsymbol{A}$: $[\boldsymbol{Q},\boldsymbol{R}] = \text{qr}(\boldsymbol{A})$, where $\boldsymbol{Q} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{R} \in \mathbb{R}^{m\times k}$
Set $\boldsymbol{Q}_A = \boldsymbol{Q}(:, 1:k)$, $\bar{\boldsymbol{Q}}_A = \boldsymbol{Q}(:, k+1:m)$, $\boldsymbol{R}_A = \boldsymbol{R}(1:k, :)$
Compute p optimal integer solutions: $\hat{\boldsymbol{Z}} = \text{ils}(\bar{\boldsymbol{Q}}_A^T\boldsymbol{B}, \bar{\boldsymbol{Q}}_A^T\boldsymbol{y}, p)$
Solve the following upper triangular equations to get the the corresponding p real solutions:

$$\boldsymbol{R}_A\hat{\boldsymbol{X}} = \boldsymbol{Q}_A^T(\boldsymbol{y}\boldsymbol{e}^T - \boldsymbol{B}\hat{\boldsymbol{Z}}), \text{ where } \boldsymbol{e} = [1, \cdots, 1]^T \in \mathbb{R}^p$$

References

[1] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger. *Closest point search in lattices*. IEEE Transactions on Information Theory, 48:2201–2214, 2002.

[2] Å. Björck. *Numerical Methods for Least Squares Problem*. Philadelphia: SIAM, 1996.

[3] X.-W. Chang, X. Yang, and T. Zhou. *MLAMBDA: A modified LAMBDA method for integer least-squares estimation*. Journal of Geodesy, 79:552–565, 2005.

[4] H. Cohen. *A Course in Computational Algebraic Nunber Theory*. Springer-Verlag, Berlin, Germany, 1993.

[5] P.J. De Jonge and C.C.J.M. Tiberius. *LAMBDA method for integer ambiguity estimation: implementation aspects*. In Delft Geodetic Computing Center LGR-Series, No.12, 1996.

[6] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, 3rd edition, 1996.

[7] A.K. Lenstra, Jr.H.W. Lenstra, and L. Lovasz. *Factoring polynomials with rational coefficients*. Mathematische Annalen, 261:515–534, 1982.

[8] C.P. Schnorr and M. Euchner. *Lattice basis reduction: improved practical algorithms and solving subset sum problems*. Mathematical Programming, 66:181–199, 1994.

[9] P.J.G. Teunissen. *GPS carrier phase ambiguity fixing concepts*. In P.J.G. Teunissen and A. Kleusberg, editors, *GPS for Geodesy*, pages 317–388. Springer-Verlag, New York, 2nd edition, 1998.

[10] T. Zhou. *Modified LLL Algorithms*. Master's Thesis, McGill University, School of Computer Science, 2006.