SLR_Homework

Exercise 1.2:

Because A and x are known, so problem is $argmin_{\Pi \in P} ||y - \Pi c||_2$, It is mean I want $||y - c_\Pi||_2^2$'s answer is minimum.
$||y - c_\Pi||_2^2 = \Sigma_1^m (y_i - c_{\Pi i})^2 = \Sigma_1^m (y_i^2 + c_{\Pi i}^2) - 2\Sigma_1^m y_i c_{\Pi i}$. By sequence inequality, we know when When order is order， $\Sigma_1^m y_i c_{\Pi i}$ is maximum，so the answer is minimum.

(Unique) Because out of order is smaller than bigger, and In Probability $c_{\Pi i} == c_{\Pi j}$ is zero probability event, In computer, I think it is a impossible event by use matlab try lots of times, so the solution is unique.

```
Algorithm 1
Input y, A, x
Output Pi
Start:
  sort(y) and record their origin location s1
  sort(c) and record their origin location s2
  Generate Pi from s1 and s2
End
```

so the complexity is $O(mlogm)$.

Exercise 2.1:

（1）It easy to knows $y = Ax^*$ by least-squares, so can get minimum by $x^* =$（$A^T A)^{-1} A^T y$.

(2) the error is equal $||(A^TA)^{-1}A\zeta||/||x||$, Generally, when n = 1 and m = 1, then the question is to know $|(A^{-1}\zeta)|/|x| \le 1/100$, $E(\zeta) = \delta\sqrt{2/\pi}$, $E(A) = E(x) = \sqrt{2/\pi}$, so we can know $\delta \le \sqrt{2/\pi}/100$, the error's exception is less than 1%.(I try use code to count when sigma = 0.0050, the error is less than 1%).

Exercise 2.2:

(2)Complexity, I don't know matlab's full permutation algorithm. But I know use next_permutation(A C++ function) can get all permutations in $O(m^2)$. and get x_hat's complex is $O(m*n^2+n^3+n^2*m)$, so the complex is $O(m!*(m*n^2+n^3+n^2*m))$. Generally speaking, the computer counts 1e8 times in one second, and 1e11 times in one hour.

(3)So when $n^2m!(m+n) < 1e11$, the algorithm will to be effcient. And the algorithm is equal to Exercise 2.1, so $\delta \le \sqrt{2/\pi}/100$ can get small error(I try use code to count when sigma = 0.0049, the error is less than 1%).

(4)And the algorithm can toperate 100% shuffled data.

Exercise 3.1:

(2)

```
Given:
    A - a Model parameters from y = PAx + zeta
    y - function's output, a set of observations

Return:
    bestFit — data parameters which best fit the function

minerror = inf
bestFit = null
XSet = null
ybar = random subvector in Rn of y

for Ai in all n*n matrixes formed by the rows of A {
```

```
  xi = Ai's inverse matrix * yi
  add xi point into XSet
}

for xi in XSet {
  Pi = SLR_1_Pi_given_x(A, y, xi)
  error = norm(y - PiAx)
  if error < minerror {
    minerror = error
    bestFit = xi
  }
}
return bestFit
```

(3)Algorithm has $m!/(m-n)!$ times iterations, and each iterations, compute inverse matrix is $O(n^3)$, other matrix operation is $O(n^2)$ can be ignored, because $m > n$, so PiA is $O(m*n)$, I don't ignore it.

The complex is $O((m!)/(m-n)!*(n^3+n*m))$. Moreover $m > n$, I write complex is $O(n*m!*(n^2+m)/(m-n)!)$.

(4)

$m = \{20, 40, 60, 80, 100, 120, 140, 160, 180, 200\}$

$n = \{6, 5, 4, 4, 4, 3, 3, 3, 3, 3\}$

(5)

```
errorsRS=
    0.0394

errorsBF=
    0.0119
```

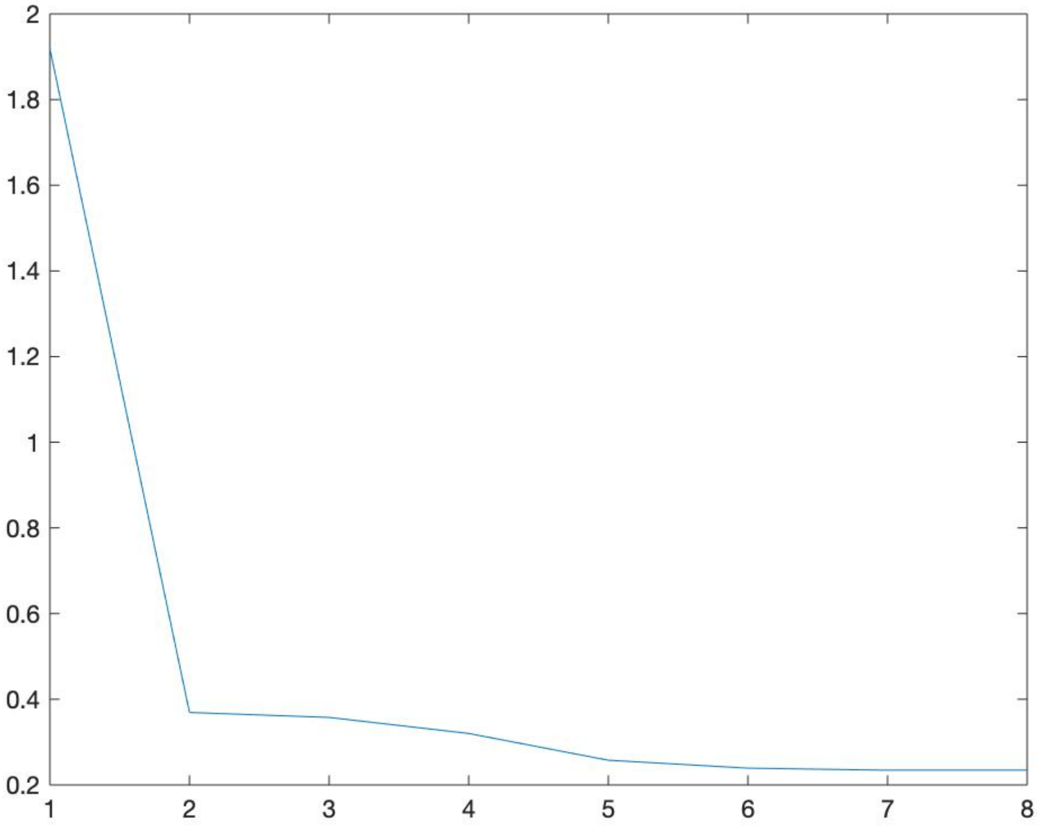
According the answer, RANSAC produces larger errors, According the exercise 2.2, I know errors is influenced by ζ. when the algorithm is Brute force, x's answer is influenced by more ζ's parameters. Although their exception are same, but bacause sample's number, so RANSAC's variance will be bigger.

Exercise 3.2

For a subvector y, If I want pick the corract submatrix one times. It is $1/A_m^n$ probability. So after T times can't pick it the probability is $1-\frac{1}{A_m^n}=(\frac{m!-n!}{m!})^T$, It mean after T times, The probablity that pick it at least one time is $p=1-(\frac{m!-n!}{m!})^T$, and I want $p>\gamma$. Then the inequality can be transfered to $T*ln((m!-n!)/m!)<ln(1-\gamma)\rightarrow T>\frac{ln(1-r)}{ln((m!-n!)/m!)}\rightarrow T>log_{(m!-n!)/m!}(1-r)$

Exercise 4.1

(1)



(2)

$$\Pi_{k+1} = \underset{\Pi \in P}{argmin} ||y - \Pi A\mathbf{x}_k||_2 \quad (1)$$
$$\mathbf{x}_{k+1} = \underset{\mathbf{x} \in \mathbb{R}^n}{argmin} ||y - \Pi_{k+1} A\mathbf{x}||_2 \quad (2)$$
$$k+1 \quad object \quad function: ||y - \Pi_{k+1} A\mathbf{x}_{k+1}||_2 \quad (3)$$
$$k \quad object \quad function: ||y - \Pi_k A\mathbf{x}_k||_2 \quad (3)$$
$$(2) \leq ||y - \Pi_{k+1} A\mathbf{x}_k||_2 \leq (3) \qquad (4)$$

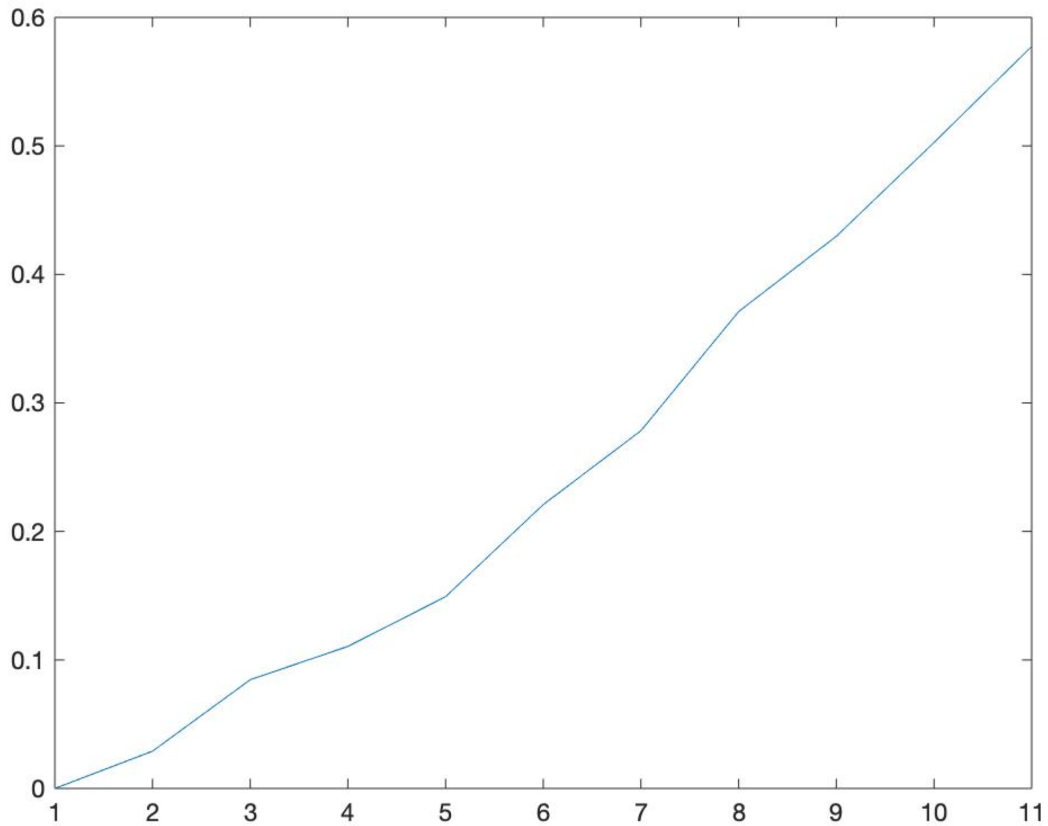
so object function decreases in each iteration.

(3)

The algorithm's each iteration is to get the minimum $\mathbf{x}_{k+1}$, so when $\mathbf{x}_{k+1} \approx \mathbf{x}_k$, I think the algorithm should be terminate. so the condition is $||x_k - x_{k+1}||/||x_{k+1}|| < 1e^{-16}$, I use the error, because the numerical error about double is about $2e^-16$.

(4)

The algorithms's error about stuffled_ratio:



Exercise 4.2

EM algorithm:

EM's goal is

$$\theta, z = \underset{\theta,z}{argmax}\, L(\theta, z) = \underset{\theta,z}{argmax}\, \Sigma_{i=1}^{m} log \Sigma_{z^{(i)}} P(x^{(i)}, z^{(i)}|\theta) \tag{1}$$

z is a latent variable, θ is parameters of the distribution in probability. x is sample's state.

It's difficult to get (1)'s derivate, so use inequality to move $\Sigma_{z^{i)}}$ to outside. Then,

$$\Sigma_{i=1}^{m} log \Sigma_{z^{(i)}} P(x^{(i)}, z^{(i)}|\theta) = \Sigma_{i=1}^{m} log \Sigma_{z^{(i)}} Q_i(z^{(i)}) \frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})} \geq \Sigma_{i=1}^{m} \Sigma_{z^{(i)}} Q_i(z^{(i)}) log \frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})} \tag{2}$$
$$\Sigma_z Q_i(z) = 1, \quad 0 \leq Q_i(z) \leq 1$$

We hope maximize the lower bound. Assume θ is known. than the equation value is deponds on $Q_i(z)$ and $p(x^{(i)}, z^{(i)})$. If we hope $\geq$ become $=$. Because Jansen indequlity, We have $\frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})} = c$

$$P(x^{(i)}, z^{(i)}|\theta) = cQ_i(z^{(i)})$$
$$\Sigma_z P(x^{(i)}, z^{(i)}|\theta) = c\Sigma_z Q_i(z^{(i)})$$
$$\Sigma_z P(x^{(i)}, z^{(i)}|\theta) = c$$
$$Q_i(z^{(i)}) = \frac{P(x^{(i)}, z^{(i)}|\theta)}{c} = \frac{P(x^{(i)}, z^{(i)}|\theta)}{\Sigma_z P(x^{(i)}, z^{(i)}|\theta)} = \frac{P(x^{(i)}, z^{(i)}|\theta)}{P(x^{(i)}|\theta)} = P(z^{(i)}|x^{(i)}, \theta).$$

When $Q_i(z^{(i)}) = P(z^{(i)}|x^{(i)}, \theta)$. Maximizing $\Sigma_{i=1}^{m} \Sigma_{z^{(i)}} Q_i(z^{(i)}) log \frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})}$ is maximize $L(\theta)$. So EM's step is, first fixed θ, count $Q_i(z^{(i)})$, and then count $argmax_\theta\, \Sigma_{i=1}^{m} \Sigma_{z^{(i)}} Q_i(z^{(i)}) log \frac{P(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})}$.

When we use the idea to the problem of linear regression without correspondences. It's same. We don't know Π and $\mathbf{x}$. we want $min||y - \Pi A\mathbf{x}||$, so we have $||y - \Pi Ax|| \leq ||y - \Pi^* Ax||$, $\Pi^* = f(x)$, Because Exercise 1, I know Π is a function's answer about x, so It as same as EM, so when I assume x, I can get Π, and then get the proper x's answer is to minimize the upperbound.

Exercise 5.1

(1) My *brute force* solution is select all the x rows (x from n to m) of A, assuming that these rows match, then solve x, add the solution set, compare all L0−norm values, and choose the minimum value as the answer.

Origin x can the minimum from $||y - Ax||_0$, $[x^T, -1]^T$ is belong to $N([X', -1])$, X' is fromed by inlier data. because X is random, so can't find other point belong in inlier data's range base. Moreover inlier data is more than 50%, so other x^*'s answer is smaller than origin x.

So the algorithm is make sense.

(2)

Refer to code SLR_6_CVX.m. the solution isn't sensitive to the choice of λ. And when sigma is zero, error of the method which is noiseless is always little. By change sigma to a large value, The noisy methd's error is about as same as little sigma. By 100 averages, So I can think the I'mplementation is right.(the photo's value should be divided by 100)

```
------------------------------------------------------------
Status: Solved
Optimal value (cvx_optval): +0.00470729

   14.2721

   35.9533
```

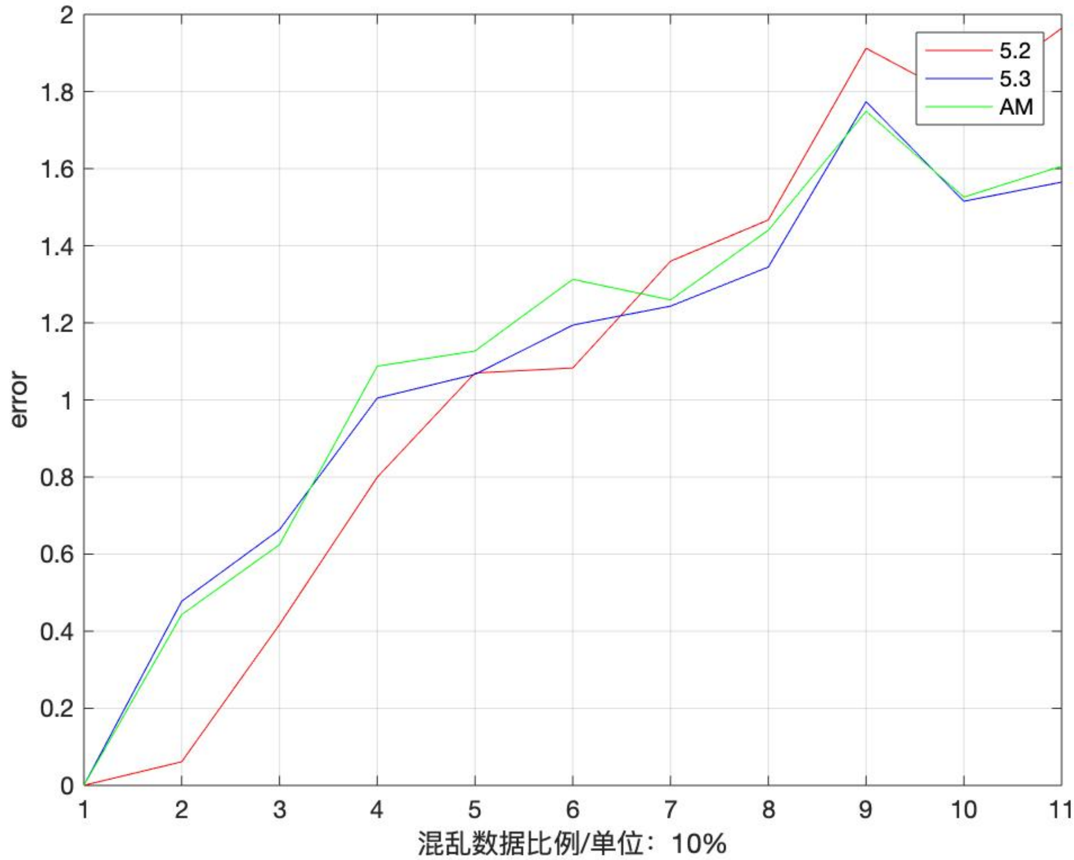
(3)

Because treat mismatches as outliers and the rest as inliers, and $y - Ax = e$, so (5.3)'s part 2 is equl to (5.2), and the part 1 is to be estimate origin e's value.

(4)

When m and n is too large, the algorithm run long time. and if $m >> n$, the application appear some bug. but when m = 1000 and n = 200. the algorithm is correct. so I guess It correct, when m = 10000.

(5)



Exercise 6.1

(1)

$p_2(Ax) = 4(21x^2 + 47xy + 27y^2)$

$p_3(Ax) = 496x^3 + 1686x^2y + 1932xy^2 + 744y^3$

(2)

the error is about $9.3071e-13$.

(3)

when $m = 100$, about 0.1 seconds.

```
历时 0.130337 秒。
   3.0551e-16
```

when m = 10000, about 2.3 seconds.

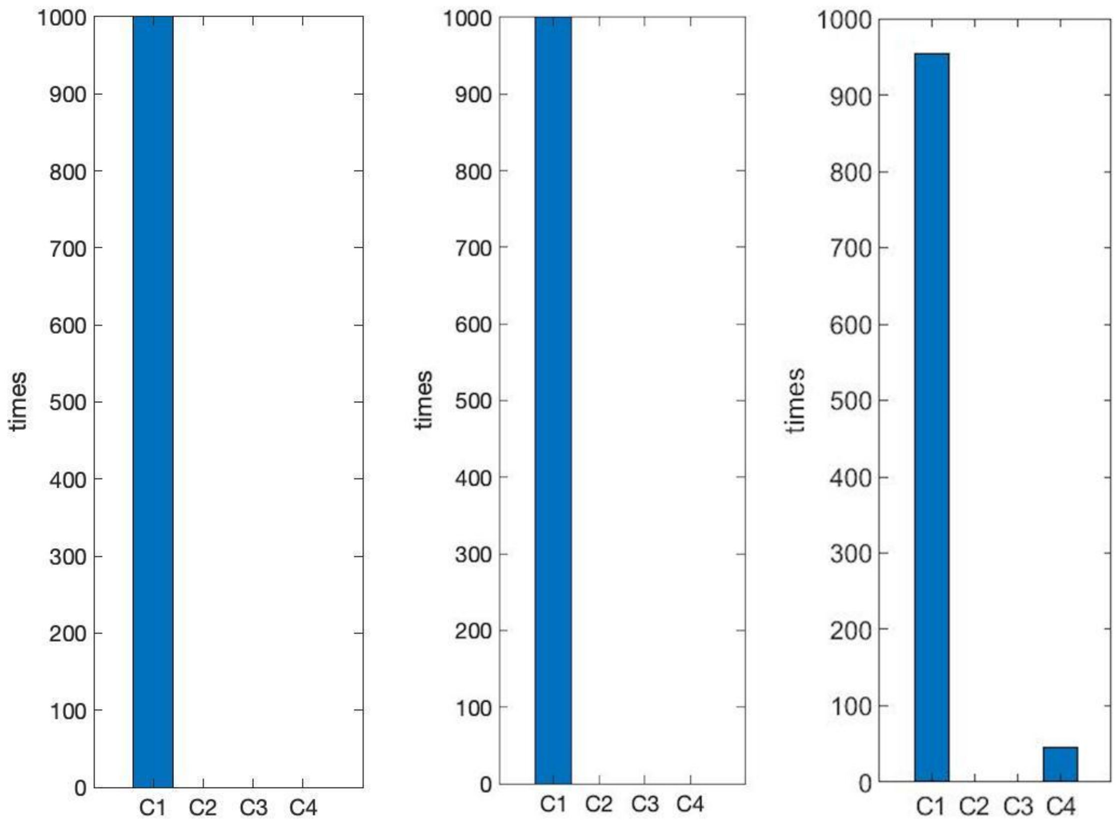
```
历时 2.412763 秒。
   6.0147e-16
```

when m = 50000, about 11 seconds.

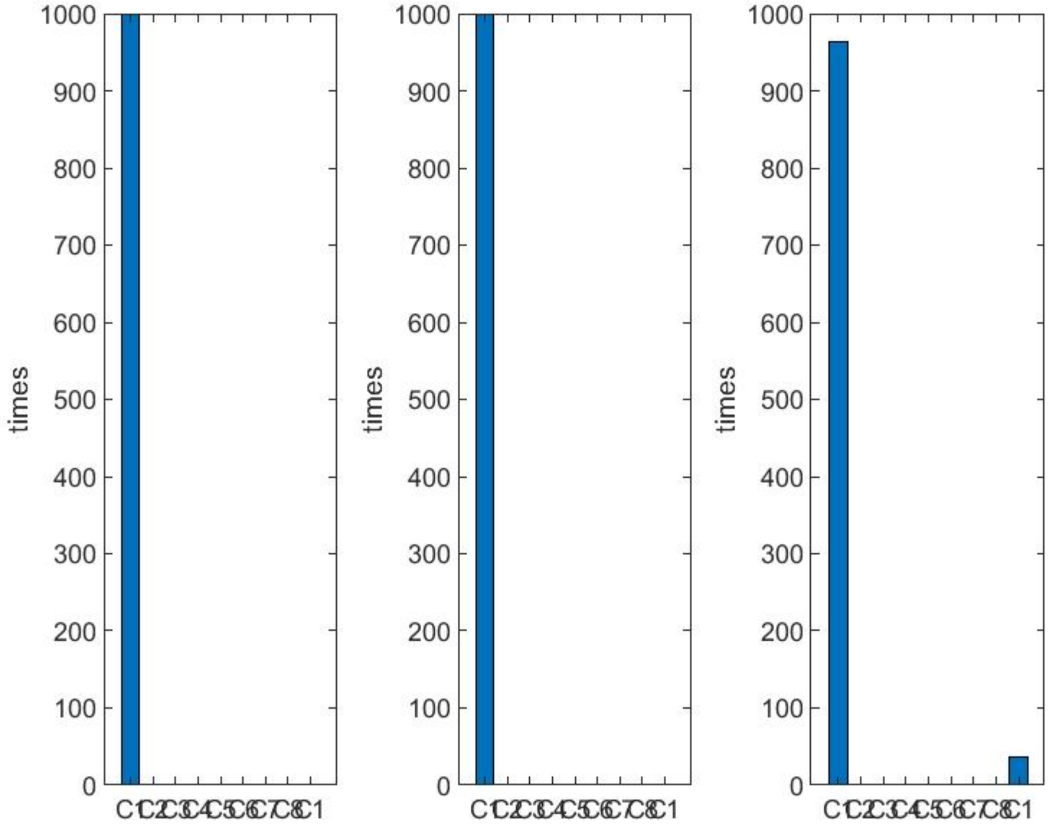
```
历时 11.156383 秒。
   9.1461e-16
```

emmm, I'm just curiosity about why the difference between n2's setup_elimination_templete function and n5's.

(4)



n=3 n=4 n=5



When I run the program, I find when $n = 5$, the rank is not full, so sometimes the error is bigger. But the problem is from solver_SLR_n5.m, so I think it's reason is when n = 5, the data's matrix isn't full rank in column.