Near-singularity of Gaussian Correlation Matrices

Harshvardhan

June 2017

Project: This project is about near-singularity of Gaussian correlation matrices, i.e. following class of matrices

$$R_{ij} = R(x_i, x_j) = \prod_{k=1}^{d} \exp(-\theta_k |x_{ik} - x_{jk}|^2).$$

To understand the following for Gaussian correlation matrices:

- Near-singularity and it's measurement
- Conditions, frequency of it's occurrence and it's dependency on
  1. Design size, n
  2. Input dimension, d
  3. Parameter, θ
- Getting rid of this near singularity without compromising much of accuracy.

1 Near-singularity and it's measurement

Near singularity is a concept by which we can predict how reliable our solutions of a given matrix equations are. Like, if we have a matrix equation $Ax = b$, then if A is singular then there isn't any solution of the equation. But, *whatever is impossible at zero, would be difficult near it.* So, for near singular matrix, we need to take care that the solutions are reliable. It is very much possible that the matrix which we have has all small values as it's elements, hence a small determinant. But often, these matrices will have very large inverses, and hence the solution won't be reliable.

To define how much the the given matrix is near singular, we will use *condition number.* [1] First, we will have to find the order of the condition number. Let's say, it is of the order k. [2] Then, find the number of significant digits in the matrix obtained, i.e. R and say that to be r. Then the matrix is said to be *near-singular* if $r - k \leq 0$. In real life applications, we will have to find the significant numbers of the original correlation matrix case-by-case. We have assumed that on calculation we will have r significant figures and then we proceed to find the near-singularity conditions of that particular matrix by the method we described.

[1] *Condition number* is the ratio of the largest singular value and the smallest singular value of a given square matrix.
[2] When a number is written as $r \times 10^k$ where $0 \leq r \leq 1$ then, it is said to be of the order $k - 1$.

Another method to find if a matrix is near-singular is by checking relative error [3]. in it's solutions by altering the elements of the matrix by a small δ, where δ is the relative error of the matrix R. If we change the elements of R by a small proportions, say δ and observe the relative change observed in the solution of the matrix equation $Ax = b$. For simplicity, consider only linear equations are present in $Ax = b$. So, theoretically, if the matrix is non-singular the error in the solution also should be δ, practically somewhere very close to δ. And if the solutions observed after changing the matrix from A to $A+\Delta A$, the solution change to somewhat different from $x+\Delta x$, the matrix can be said near-singular. But this method of identification is a bit troublesome and costly in calculations since we will have to solve multiple equations.

We can neglect the *determinant method of identification* of near singularity as it is well covered in the definition of condition number itself, $cond(A) = ||A||||A^{-1}||$.

2 Conditions and frequency of it's occurrence

A matrix can be said to be near singular when order of the condition number and the number of significant digits in the entries of the matrix are equal (*see section 1*). The conditions in numerical computations are more easier as `double` data type which usually has 14 digits of precision, so in mathematical computations using `double` numbers, whenever the order of the condition number is ≥ 14, then it is said to be near singular with non-accurate solutions. Also, when r and k are both large and *nearly* the same, then the matrix can be called near singular. But this definition is completely based on the machine's precision of `double` data type. So, a better method to understand the near-singularity is to understand it in absolute terms using condition numbers and then relate the largeness of condition number to near-singularity.

The matrix depending on design size n, input dimension d and parameter p, was computed and then the arguments n, d and p were subsequently changed and the results are as below.

2.1 Dependence on design size n

With the increase in design size n, there was a increase in the largeness of the condition number. As n grew up from $n = 10$ to $n = 100$ with the breaks of 10, there was an evident increase in the order of condition number and hence of course condition number, itself. Although direct proportionality could not be observed between *near-singularity* and design size, n, but seeing the increasing size of condition number with increasing n one can easily deduce the increasing *near-singularity* and hence some proportionality. Other arguments for generation of Gaussian correlation matrices, θ and dimension d, were kept constant at $\theta = U(0, 1)$ and $d = 10$.
Following algorithm was used to test dependence on n:

[3]Relative error in y is defined as $\frac{\Delta y}{y}$

```
Data: $z$ will store orders of 500 Gaussian correlation matrices generated using GCM(), which is
      a user defined function. order() returns the order of the numeric value passed and $\kappa()$
      returns the condition number of the matrix passed as the argument. hist() displays
      histogram of $z$ for each tried $n$. Subsequently, ten histograms were generated.
Result: Vector $z$ containing order of condition numbers of the 500 Gaussian correlation
        matrices $R$
$f = 1$;
while $f \leq 10$ do
    $n = 10 * f$;
    $d = 10$;
    $\theta = U(d, 0, 1)$;
    $i = 1$;
    while $i \leq 500$ do
        R = GCM(n,d,$\theta$);
        z[i] = order($\kappa(R)$);
        $i = i + 1$;
    end
    hist(z);
    $f = f + 1$;
end
```

Algorithm 1: Calculating order of Condition Number with changing n

All the histograms observed on changing $n = 10$ to $n = 100$ are appended at the end of the document. The graph of increasing mean order of condition number with respect to design size n is also placed at the last.

So, as observed, with the increase in design size, there is a increase in condition number and hence subsequent increase in cases of near-singularity. Higher dimensions will lead to more near-singular Gaussian correlation matrices and lower ones had much lesser chances to be near-singular.

2.2 Dependence on input dimension d

With the increase in the dimension size d, there was a visible decrease in condition numbers of the matrix with no exceptions. Dimension sizes from $d = 5$ to $d = 50$ were tested. To the extent when $d = 50$, the order of condition numbers reduced to one again without any exceptions. In fact, the orders of condition numbers were observed in a decreasing fashion like shown in the figure appended. Other arguments for generation of Gaussian correlation matrices, θ and design size n, were kept constant at $\theta = U(0, 1)$ and $n = 10$.

Following algorithm was used to test dependence on n:

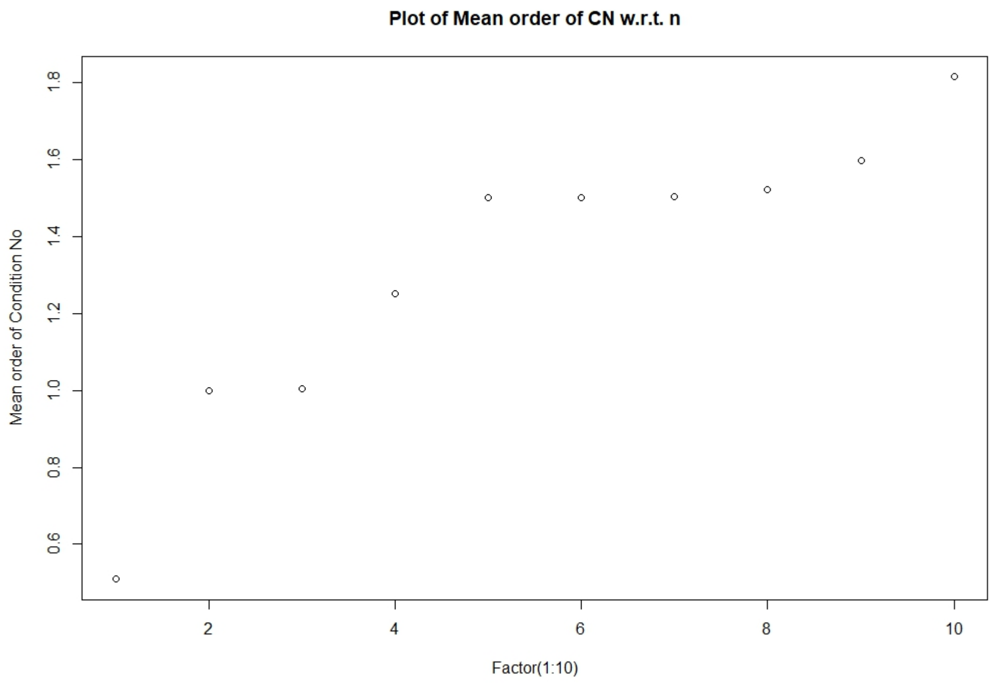


Figure (1) Change in order of condition numbers for 100 simuations for 10×10 matrix with changing design size from 10 to 100.

```
Data: $z$ will store orders of 500 Gaussian correlation matrices generated using GCM(), which is
      a user defined function. order() returns the order of the numeric value passed and $\kappa()$
      returns the condition number of the matrix passed as the argument. hist() displays
      histogram of $z$ for each tried $d$. Subsequently, ten histograms were generated.
Result: Vector $z$ containing order of condition numbers of the 500 Gaussian correlation
        matrices $R$
$f = 1$;
while $f \leq 10$ do
    $n = 10$;
    $d = 5 * f$;
    $\theta = U(d, 0, 1)$;
    $i = 1$;
    while $i \leq 500$ do
        R = GCM(n,d,$\theta$);
        z[i] = order($\kappa(R)$);
        $i = i + 1$;
    end
    $hist(z)$;
    $f = f + 1$;
end
```

Algorithm 2: Calculating order of Condition Number with changing d

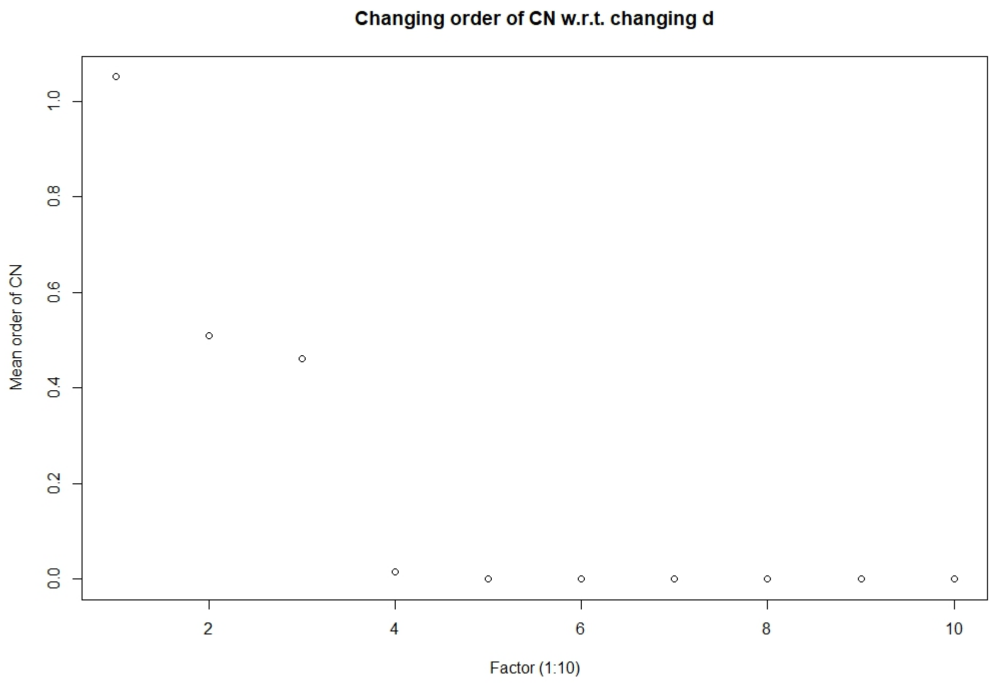


Figure (2) Change in order of condition numbers for 100 simuations for 10×10 matrix with changing dimension size from 5 to 50.

We tried for the order of condition numbers with increasing condition numbers from $d = 5$ to $d = 50$. As observed, with the increase in dimension size, there is a decrease in condition number and so decrease in cases of near-singularity. Higher dimensions lead to more non-singular Gaussian correlation matrices and lower ones had more chances to be near-singular. When dimension size was increased to 4, there was no case of near singularity observed for design size upto 50. So, as a rule of thumb, one can say that for design size less than 50 and dimension greater than 4, there are no near singularity cases observed.

2.3 Dependence on parameter θ

Parameter θ is an important element in the formation of Gaussian correlation matrices. As you can see, it is directly multiplied by the squared-difference which means that it should have important effect in it's generation and should affect near singularity, greatly.
We tried three parameters to study their effect on the generated matrices. The parameters tested were:

- $\theta = \theta_i = (0, \infty)$,
- $\theta = \frac{1}{\lambda_i}$ where $\lambda_i = (0, \infty)$ and
- $\theta = 10^{\beta_k}$ where $\beta_k = (-\infty, \infty)$.

Let's discuss each of the parameter in detail.

2.3.1 $\theta = \theta_i = (0, \infty)$

In this case, we started with smaller values of θ and then increased the value of θ to understand the effect of theta on condition number of the matrices generated. Like previously, we tried to compute and compare the order of the condition numbers of the matrices. We observed that condition numbers were getting smaller with respect increase in the parameter from $\theta =$ `0,20` to $\theta =$ `0,200`. In fact, with θ as large as 200, the condition numbers were of negative orders.

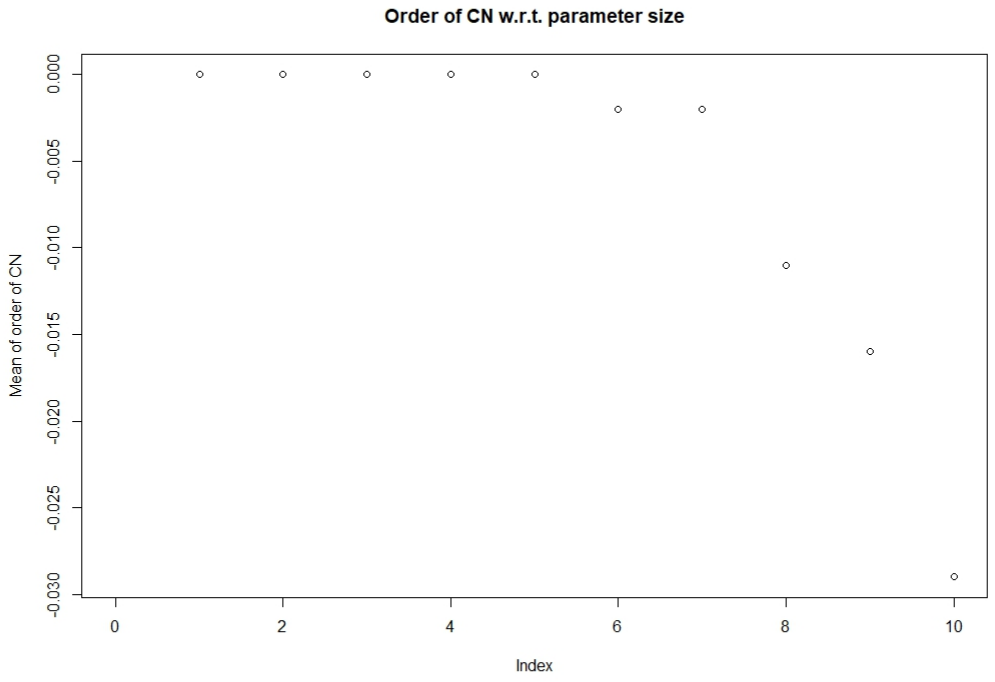


Figure (3) Order of condition numbers with increasing parameter size θ from `U(0,20)` to `U(0,200)`, for parameter $\theta = \theta_i = (0, \infty)$.

2.3.2 $\theta = \frac{1}{\lambda_i}$ where $\lambda_i = (0, \infty)$

Like last case, in this case too we started with smaller values of λ and then increased the value of λ and observed the effect on the order of the condition number of the generated Gaussian correlation matrix. We started with factor of $f = 1, i.e.\lambda = U(0, 20)$ and went upto $f = 20, i.e.\lambda = U(0, 400)$. We observed that as the value of λ increased the order of the condition number also increased.

This table concludes the effect on near-singularity by various arguments n, d and θ.

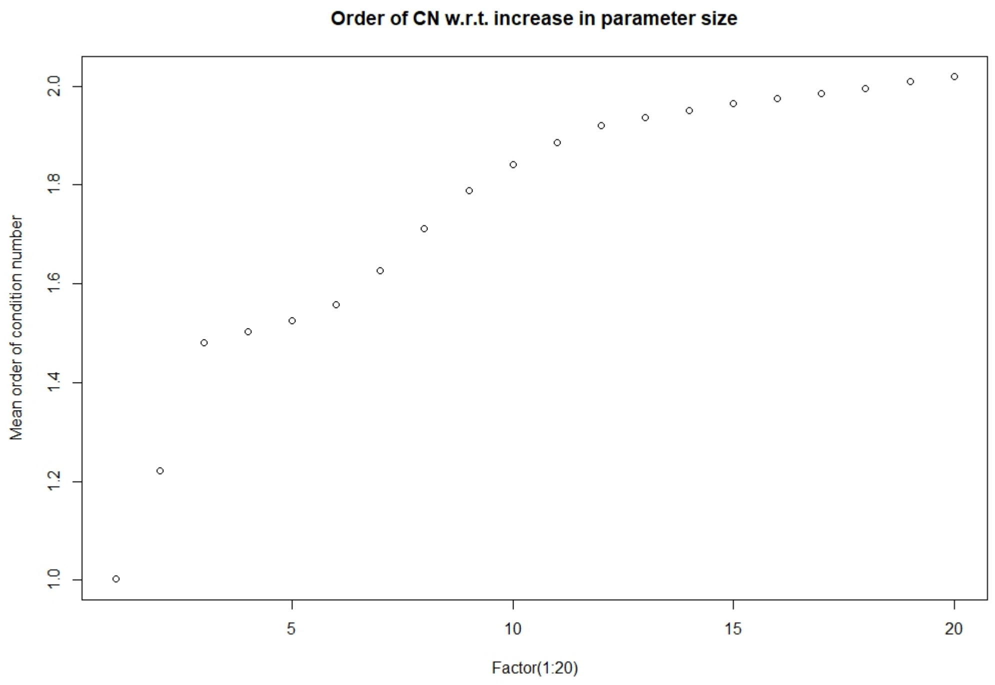


Figure (4) Order of condition numbers with increasing parameter size λ from `U(0,20)` to `U(0,400)`, for parameter $\theta = \frac{1}{\lambda}$.

Serial No.	**Argument**	**Change in CN with increase in argument**	**Near-singularity cases**
1	Design Size n	Increases with increase in n	More near-singular cases
2	Dimension d	Decreases with increase in d	Lesser near-singular cases
3(a)	Parameter, $\theta = (0, \infty)$	Decreases with increase in upper limit of interval for θ	Lesser near-singular cases
3(b)	Parameter, $\theta = \frac{1}{\lambda}$, $\lambda = (0, \infty)$	Increases with increase in upper limit of interval for λ	More near-singular cases

3 Getting rid of near-singularity

Near-singularity can be a problem. As already told in part one, *whatever is impossible at zero, would be difficult near it.* We usually try do decrease the near-singularness of a matrix by pulling it's determinant up or pushing it's condition number down and they are essentially the same. Let's see how can we alternatively solve the equation problem $Ax = b$ so that *singularity or near-singularity* is not a problem. In some of these methods, we try to analyse the errors that could be there. For this, let's say we have two equations $AX = b$ and $\tilde{A}\tilde{x} = b$, with b not equal to zero. Now, it can be proved that

$$\frac{||\tilde{x} - x||}{||x||} \leq \kappa(A)\frac{||\tilde{A} - A||}{||A||}.$$

So, the *maximum relative error* is $\frac{||\tilde{x}-x||}{||x||}$. We will use this while computing the errors when we try generating a new matrix from the old one.

3.1 Solving using Generalised Inverse

One way to solve it is using **Generalised Inverse** or **Psuedoinverse**. The good part about psuedoinverse is that they are equal to the actual inverse if inverse exists. [4] Now, using the *psuedoinverse* we can solve the matrix equation $Ax = b$.

1. Perform pre-multiplication on both sides by the *psuedoinverse* of the matrix A (call it A^g).
2. So, we have $A^g Ax = A^g b$.
3. Now, when A^g is psuedoinverse, then $A^g A$ is *identity* matrix. So, the solution now is $A^g b$.

This solution obtained isn't exactly accurate but has *least errors possible.* [5]

In this computation, we used the conditions of generalised inverse for checking how reliable it is. We found that, out of 150000 computations, $||RR^{-1}R|| = ||R||$ this was *false* in 145452 cases; and $||R^{-1}R|| = 1$ was *false* in 147633, making a difference of 1.45% cases. So, one of this condition could be safely removed so as do reduce computation time.

3.2 Solving using eigenvalue decomposition

The eigenvalue decomposition of a matrix breaks a matrix as a product of vector and its transpose with corresponding eigenvalues as weights. If the inverse is to be calculated, then these weights change to their reciprocals. Now, since we need to choose special boundary value to select the eigenvalues that could be used as weights. We try for some values and try analysing the errors that would appear upon choosing any particular boundary value.

We calculated following things:

- Condition number of the original matrix,
- $||RR^{-1}R||_2$,
- $||R||_2$,
- $R*$, which was reconstructed using $\sum_{i=1}^{n*<n} ee^T \lambda_i$,
- R^{-1} was reconstructed using $\sum_{i=1}^{n*<n} ee^T \frac{1}{\lambda_i}$,

[4] A $n \times m$ matrix A has a generalised inverse G such that $AGA = A$.

[5] By least error we mean *ordinary least square errors.*

- Sum of original eigenvalues and sum of $n* < n$ eigenvalues,
- LU decomposition and reconstruction and error therein in the original matrix and the reconstructed matrix.

3.3 Solving using Singular Value Decomposition

3.4 By introducing a small nugget in all the elements

In this method, we try to introduce a small nugget (or an error), δ, which is added to all values of the matrix, i.e. $R' = R + \delta J$, where $J_{ij} = 1, 1 \leq i, j \leq n$.

3.5 By introducing a small nugget in diagonal elements

3.6 By introducing a small relative nugget in all the elements

3.7 Proposition: Reducing the condition numbers by dividing highest ordered numeric value

Let's talk about how can we bring the condition number down. In this method we can actually bring condition numbers down by a large order. Out of 100 simulations carried out of 100 matrices which had large condition numbers (all were artificially generated to have large conditional numbers), the condition numbers after processing the algorithm brought down the order of condition number by about 3 when the highest condition number was of order 5. The histogram of the decrease in the order of condition numbers is at the last.

Algorithm for decreasing the condition number: Following is the algorithm used to decrease the condition number of the original matrix by multiplying row(s) with suitable number(s).

```
Data: $A$ stores a matrix $n \times n$, which has some *irregular* values, either very large or very small
      in comparison to the other values of the matrix. order() is a function which returns
      order of the number passed.
Result: $A'$ which has all equation-solving properties of $A$ but has much lower
        condition number.
$i = 1$;
while $i \leq n$ do
    $x = \max_{1 \leq j \leq n} A_{ij}$;
    $k =$ order(x);
    $j = 1$;
    while $j \leq n$ do
        $A'_{ij} = A_{ij}/10^k$;
        $i = i + 1$;
    end
    $i = i + 1$;
end
```

Algorithm 3: Creating a new matrix A' which has lower condition number but similar equation solving properties as original matrix A.

.

Now, let's analyse various properties of the generated new matrix (A').

3.7.1 Exceptions

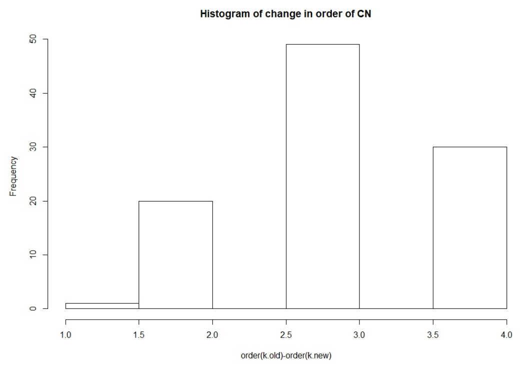
There were no big exceptions. Small exceptions, as already told, had a decrease of *one* in order of conditional number of the matrix, but all had a decrease in their condition numbers. As you can see in the histogram, just when the highest order was of 5, there was a decrease in order by about 3 with just 3 matrices out of hundred which had decrease in their order by 1, but still, all had a decrease in their orders.

3.7.2 Change in determinant

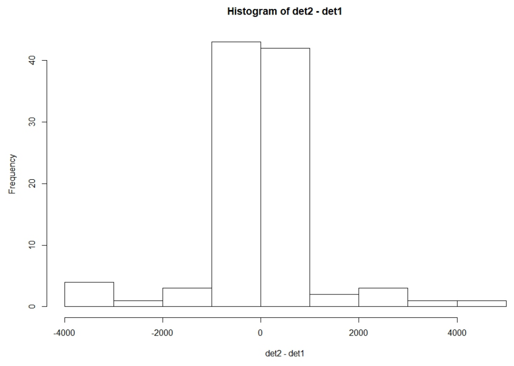
Again, after multiplying the determinant of the new matrix with suitable number (10^k, with which the rows were divided), the matrix regained it's original determinant value with somewhat error ($\approx 10^3$) or no error. The plot of the change in matrices' determinants is placed at the last.

3.7.3 LU Decomposition

LU Decomposition and reconstruction usually involves maximum errors. So, we tried to reconstruct the matrix A', found after processing through algorithm and then compare it with original A', but it generated difference and that was big (with mean ≈ 487.7). The plot for errors observed is placed at last. But an interesting observation was that some rows of the reconstructed matrix were exactly equal to the original matrix! And some values were very different. Also, the errors in LU Decomposition didn't depend on either the original condition number or the new condition number.



(a) Change in order of condition numbers for 100 simuations for 10×10 matrix with changing design size from 10 to 100 before transformation.



(b) Change in matrices' determinants for 100 artificial 10×10 matrix after transformation through algorithm in proposition.

Figure (5) Changes observed in condition numbers and determinant of 100 artificial 10×10 matrix after transformation through algorithm of proposition.

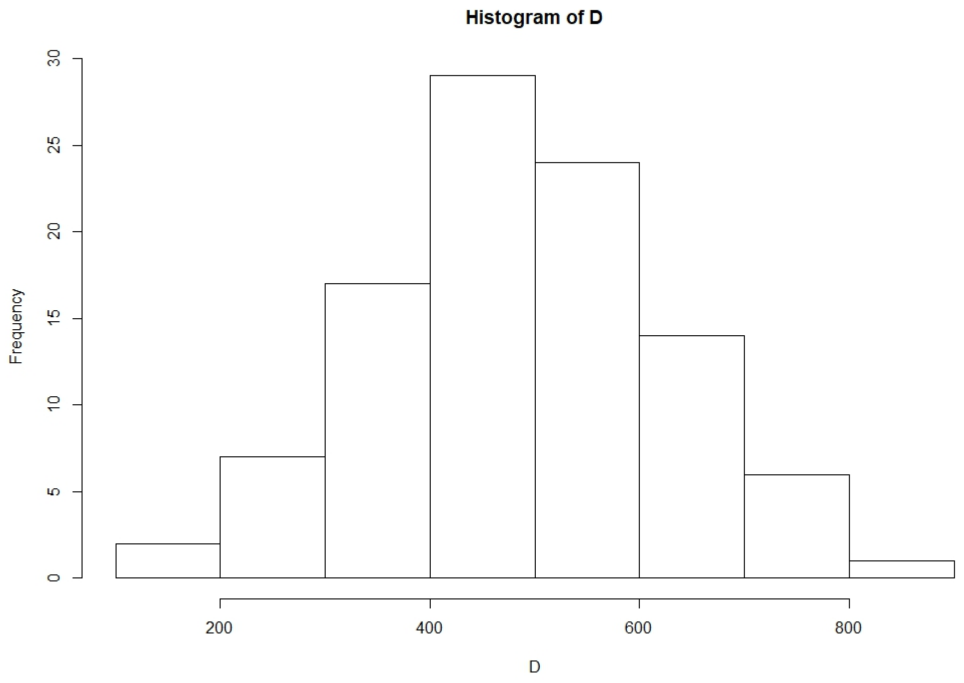
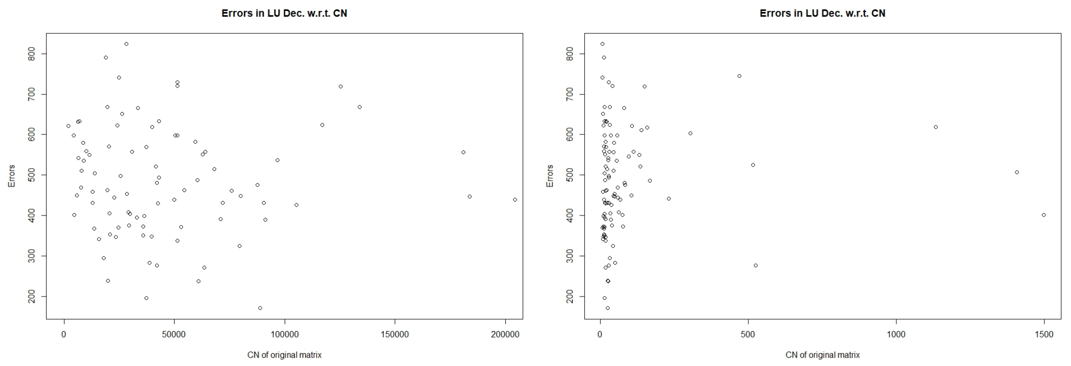


Figure (6) Errors observed in LU Decomposition of 100 artificial 10×10 matrix after transformation through algorithm in proposition.

3.7.4 Growing number of small values

After the application of the the algorithm it was found that many of the values became of very small order, it became obvious to question whether more number of large numbers of more number of small values were more dangerous for bringing matrix to near-singularity. After careful observation of the artificially generated matrices and calculation of their condition numbers, it brought us the conclusion that small values are much lesser dangerous than abnormally large values for a matrix to become near-singular.



(a) Old Condition Numbers (b) New Condition Numbers

Figure (7) Errors observed in LU Decomposition of 100 artificial 10×10 matrix after transformation through algorithm of proposition with respect to original and new condition numbers of the matrices.

3.7.5 Limitation of the proposition

The proposition is suitable only when there are large values in the matrix. If the values are small ones (like what we have in Gaussian correlation matrices), the proposition does no help. So, to conclude, we can say that even though the proposed method will be of help for many matrices, it won't really be applicable to any kind of correlation matrix as all values will be less than or equal to 1.

Acknowledgements

I throughly thank Prof Pritam Ranjan sir for giving me this chance to learn. His unending guidance has immensely enriched my knowledge in the subject. None of this would have been possible without his patience and support.