Department of Mathematics

De Rondom 70, 5612 AP Eindhoven P.O.
Box 513, 5600 MB Eindhoven
The Netherlands
www.tue.nl

Author

Maksim Kolk	(1891626)
Author 2	(ID)
Author 3	(ID)
Author 4	(ID)
Author 5	(ID)

Supervisor
Supervisor name

Date
February 10, 2025

Sparsifying the cadastral map of the Netherlands

Maksim Kolk (1891626)
`m.kolk@student.tue.nl`

Author 2 (ID)
`Author 2 email`

Author 3 (ID)
`Author 3 email`

Author 4 (ID)
`Author 4 email`

Author 5 (ID)
`Author 5 email`

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Table of contents

Title
Sparsifying the cadastral map of the Netherlands

1 Introduction

The Dutch cadastral map, in short, is a recording of admnistrative and spatial data on property and the rights involved. The map outlines land and parcel boundaries, and undergoes precision enhancements through Kadaster's program, integrating field sketches for higher accuracy. A weighted least-squares algorithm aligns common points, refining cadastral data using more accurate field-sketch points [1]. The challenge lies in managing adjustments for the 50 million cadastral points while avoiding impractical storage and computation. In order to keep the computation time manageable, we want to keep the matrix as sparse as possible. Two key questions are:

Can the covariance matrix, which gives covariances between all of the different points on the map, remain sparse for cadastral points while preserving essential properties? How can sparsification's optimality and quality be evaluated?
Several methods have been carried out and analysed during this project; we have looked at several variations on thresholding, a more sophisticated method involving perturbed matrices, and a method strictly utilizing symmetric matrix operations.

2 Problem description

We start out with a covariance matrix Σ_0. This is a square matrix containing the covariances between the included points, that is thus both symmetric and positive semi-definite. As we further require it to be invertible, we will even have it always be positive definite. After an iteration of the weighted least-squares algorithm, given by the normal equation $\left(J^T\Sigma_0^{-1}J\right)\hat{x} = J^T\Sigma_0^{-1}y$ solving for the updated coordinates $\hat{x}$ given also the Jacobian J and set of observations y, we obtain a new covariance matrix $\Sigma_u = \left(J^T\Sigma_0^{-1}J\right)^{-1}$. As we iteratively apply this algorithm (now with Σ_u instead of Σ_0), we do not want Σ_u to be much more dense than Σ_0, as this slows down the algorithm with every iteration. Our goal is thus the following:

Sparsify the matrix Σ_u to $\Sigma_s \approx \Sigma_u$, while keeping it symmetric positive definite.

The symmetric positive definiteness is required to ensure we keep an invertible covariance matrix, as this is a requirement to solve the system described by the weighted least-squares equation. Since Σ_u is symmetric by construction, we only need for all its eigenvalues to be positive in order to call Σ_u positive definite.

We investigated several methods in order to try to achieve this, all of which will be explained in further detail in the coming sections.

2.1 Metrics and notation

In order to evaluate the relative error created by a certain method, we use the following measure:

$$M(\Sigma_u, \Sigma_s) := \frac{\|\Sigma_u - \Sigma_s\|_F}{\|\Sigma_u\|_F},$$

where $\|\cdot\|_F$ is the Frobenius norm defined by $\|A\|_F := \sqrt{\sum_{i,j}|a_{ij}|^2}$. Intuitively, this measure $M(\Sigma_u, \Sigma_s)$ is an indication of the difference between the original updated matrix Σ_u and its sparsified version Σ_s. Hence, we want this value to be as small as possible. Since the diagonal elements of a covariance matrix, the variances, can be any positive real number, it might be more suitable to instead use the scaled correlation matrix, as this forces the diagonal elements to be scaled to 1, as well as the off-diagonal elements to be within the interval $[-1, 1]$ instead of any real number. This effect should be further investigated upon, though we will not touch upon this here.

Secondly, to measure the difference in sparsity between Σ_0 and Σ_s, we define:

$$R(\Sigma_0, \Sigma_s) := \frac{|\{\sigma_{ij} \in \Sigma_s \,|\, \sigma_{ij} \neq 0\}|}{|\{\sigma_{ij} \in \Sigma_0 \,|\, \sigma_{ij} \neq 0\}|}$$

Therefore, to have a functioning method, it must produce a matrix Σ_s such that $R(\Sigma_0, \Sigma_s) \approx 1$, as this implies we preserve the same sparsity as was present in the original covariance matrix Σ_0 (though a value smaller than 1 is certainly also acceptable).

Given Σ_0 and Σ_u, the challenge thus comes down to finding the Σ_s that minimizes both $M(\Sigma_u, \Sigma_s)$ and $R(\Sigma_0, \Sigma_s)$ simultaneously, which is what we will attempt to find in the coming sections.

2.2 Given data sets

For testing our methods, we were given a total of 12 covariance matrices of various sizes. One matrix, the so-called toy problem instance, had 1,847,024 non-zero entries and was 1922x1922. Throughout our testing, we noticed that the performance of all methods was substantially worse on the toy problem compared to the non-toy instances and because this instance was also highly unrealistic we decided not to use this toy problem in our results. The second and realistic data set contained 112,123 non-zero entries and was 1708x1708. We ended up having 10 extra matrices with size 1708x1708, the most sparse of which had 3158 non-zero entries and the most dense had 2,128,636. These 11 matrices combined with the toy instance were the basis for all our testing and results.

3 Creating Intuition: Thresholding

The most intuitive way to remove entries from a dense matrix is to remove the least influential values. A simple way to achieve this is using thresholding, by choosing some lower bound ε and removing all off-diagonal entries smaller than ε. This technique can quickly remove large amounts of entries while preserving the large values, and thus negligibly changing the covariance matrix. However this technique also has its downsides, as this might not preserve the positive definiteness of the covariance matrix.

As the original matrix Σ_u is symmetric positive definite, it has only positive eigenvalues, thus we assume that by removing small off-diagonal values these eigenvalues differ little and remain positive. We can however expect that at some point, removing many values might change these eigenvalues so drastically that eventually at least one of these eigenvalues turn negative, or at least non-positive. We can preview an example of what would happen if we threshold at different values of ε, and whether the positive definiteness would remain by testing. For a given example data set, this can be found in Figure 1.

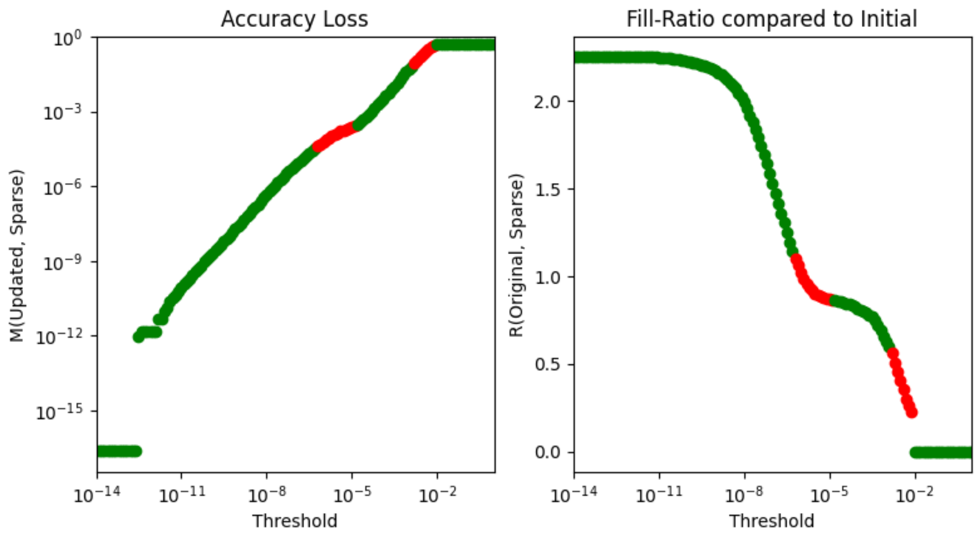

Figure 1: Thresholding on example values for $\varepsilon \in \{10^k, k \in \{-15, -14.9, \ldots, 0\}\}$. Green points indicate where Σ_s is positive definite, red points indicate non-positive definiteness.

Notice from these figures that the positive definiteness is not found on a continuous range of values for ε. Furthermore, we note that as we approach a Fill-Ratio $R(\Sigma_0, \Sigma_s)$ of approximately 1, the matrix also approaches non-positive definite sections of the graph. This indicates that just removing small off-diagonal values until the Fill-Ratio is close to 1 will generally not preserve positive definiteness. We will thus need methods that are guaranteed to, or at least expected to, preserve positive definiteness.

This simple thresholding has been applied to all 11 data sets, and similar results were found. These can be found in Appendix A. Since all data sets showed similar results, we have chosen to highlight the most "median" problem in the remainder of this paper. This was the data set where Σ_u contained approximately $1.1 \cdot 10^5$ nonzero entries. Coincidentally, this was also the first realistic data set we received.

We wish to derive an indicative system to document and visualize the performance of any method we attempt on this data set. We thus create Figure 2, in which any other method can also be mapped. This figure shows what the performance metrics of Σ_s are, thus how sparse this matrix is, and how much accuracy was lost compared to Σ_u, as explained in subsection 2.1. Note the dotted vertical line drawn at $R = 1$, which indicates whether a method performs sufficiently enough to maintain similar levels of sparsity after multiple iterations. Furthermore note the reversed axes, which were chosen to emphasize the better solutions, which now lie in the top right part of the graph. This top right area now highlights low error and relatively high sparsity.

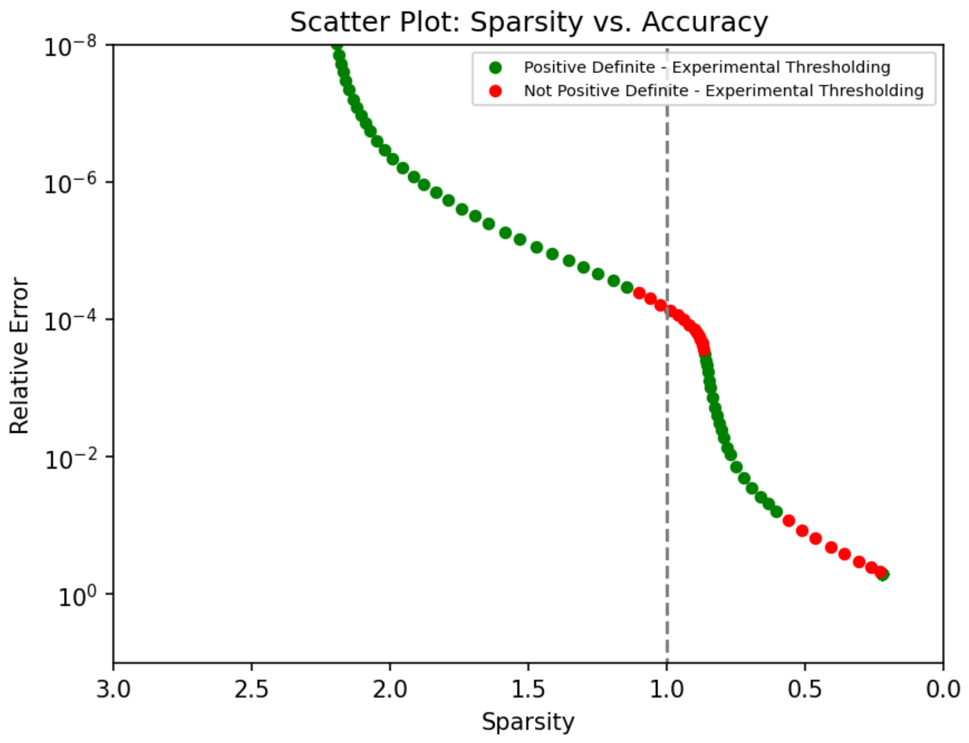

Figure 2: Results plot with thresholding. The relative error is equal to $M(\Sigma_u, \Sigma_s)$, and the sparsity is equal to $R(\Sigma_0, \Sigma_s)$

What is interesting to research is at what exact moment, and thus for what reason this immediate change occurs. Thus, for what reason does an eigenvalue become non-positive whence a certain ε is chosen? Investigating this sadly led to no concrete results. However, if this could be found, this could be very beneficial. It was notable to investigate whether we could clearly find that the smallest eigenvalue indeed turned negative in sections where the matrix was not positive definite. If structure was found in the moment these eigenvalues turned negative, logical reasoning could possibly deduce why this cutoff point occured.

From this, we observed that the eigenvalues tend to suddenly decrease just before turning negative, see Figure 3, but a structure or reasoning for why this happened was not found. We do indeed find that the smallest eigenvalue is negative whenever the matrices are not positive definite anymore.

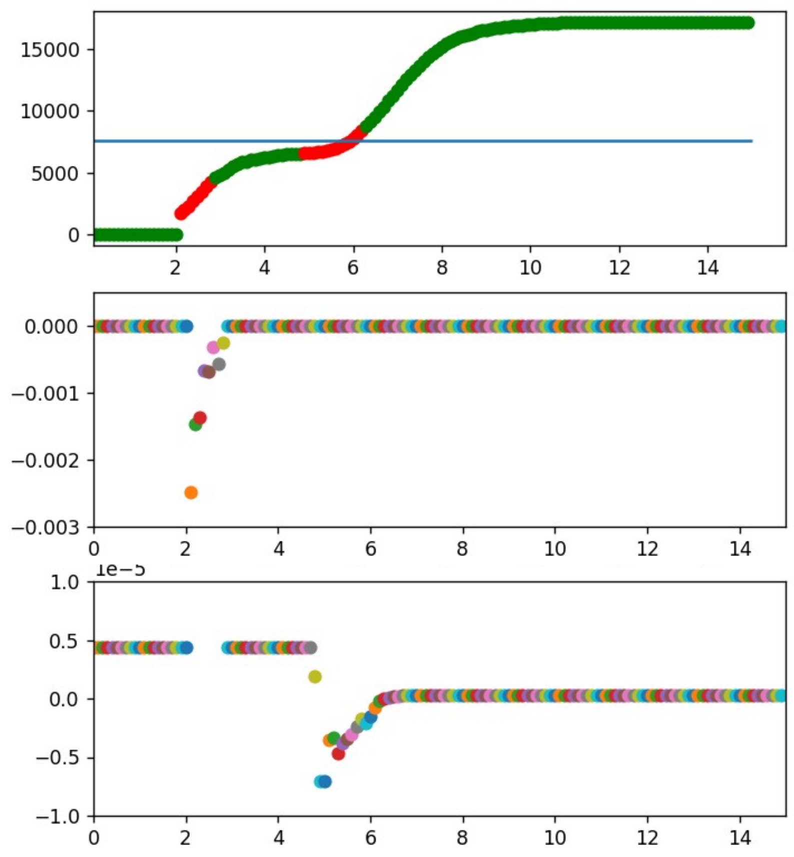

Figure 3: Eigenvalues for the thresholding model. The bottom two graphs show the minimum eigenvalue for a thresholded model, at different logarithmic scales. The first plot is similar to Figure 1 as it shows the remainig number of nonzero elements in the Matrix. The blue line represents the number of nonzero entries in Σ_0. Notice that the sections where the sparsified matrix is not positive definite, we find negative eigenvalues.

4 Method: Diagonal dominance and thresholding

In this first method, we will try to make the matrix Σ_u strictly diagonally dominant, and thereafter do thresholding. By strictly diagonally dominant, we mean:

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}| \quad \forall i, \text{ where } a_{ij} \text{ is an entry of } \Sigma_u$$

Note that we have the following for any real symmetric matrix A:

$$A \text{ is strictly diagonally dominant} \implies A \text{ is positive definite}$$

So diagonal dominance is a stronger requirement than positive definiteness.
Therefore the requirement that we have of matrix Σ_s having to be positive definite is guaranteed to be met.

The thresholding procedure will be elaborated upon in the coming sections.

4.1 Diagonal increase method

The first way in which we establish strict diagonal dominance is by carrying out Algorithm 1:

Algorithm 1 Turn Σ_u into a strictly diagonally dominant matrix Σ_s

```
Input: covariance matrix $\Sigma_u \in \mathbb{R}^{n\times n}$
Output: strictly diagonally dominant matrix $\Sigma_s \in \mathbb{R}^{n\times n}$
Initialise $\Sigma_s = \Sigma_u$
for $d = 1, ..., n$ do
  if $|(\Sigma_u)_{dd}| \leq \sum_{i=1,i\neq d}^{n} |(\Sigma_u)_{di}|$ then
    $(\Sigma_s)_{dd} = \sum_{i=1,i\neq d}^{n} |(\Sigma_u)_{di}| \; + \; 1 \cdot 10^{-16}$
  end if
end for
```

Algorithm 2 Benchmark policy

```
Input: s, Q
Output: total costs
Initialize total costs variable
for each episode do
  total costs += costs of episode
  for each store $i$ do
    if inventory of store $state$ < $s_i$ then
      order up to $\mathbf{Q}_i$
    end if
  end for
end for
return total costs
```

After performing this algorithm, we will do a thresholding procedure on matrix Σ_s with the goal of sparsifying this matrix. The thresholding procedure consists of simply choosing a certain threshold, and then setting all matrix entries that are smaller in absolute value than this threshold to zero. Of course, a smaller threshold will cause more matrix entries to be set to zero.
Note that positive definiteness of the matrix is preserved, regardless of the choice of the threshold, since the matrix is ensured to be diagonally dominant which is a stronger requirement.
Lastly, note that since this matrix is a covariance matrix, and we are only changing some but not all diagonal entries, this has the effect of altering the variance and thus the standard deviation of some variables.

4.2 Row decrease method

The second way in which we establish strict diagonal dominance is by carrying out Algorithm 3:

Algorithm 3 Turn Σ_u into a strictly diagonally dominant matrix Σ_s

```
**Input: covariance matrix** $\Sigma_u \in \mathbb{R}^{n\times n}$
**Output: strictly diagonally dominant matrix** $\Sigma_s \in \mathbb{R}^{n\times n}$
Initialise $\Sigma_s = \Sigma_u$
**for** $d = 1, ..., n$ **do**
  **if** $(\Sigma_u)_{dd} \leq \sum_{i=1,i\neq d}^{n} |(\Sigma_u)_{di}|$ **then**
    **while** $(\Sigma_s)_{dd} \leq \sum_{i=1,i\neq d}^{n} |(\Sigma_s)_{di}|$ **do**
      $(\min(\Sigma_s)_{di,i\neq d} : \ |(\Sigma_s)_{di}| \ > 0) = 0$
    **end while**
  **end if**
**end for**
```

After performing this algorithm, we will do a thresholding procedure on matrix Σ_s with the goal of sparsifying this matrix, just like before. The following graphs depict the difference between the two diagonal dominance versions on 11 instances of various sizes. Note that the graph depicts the results of both algorithms mentioned above and that no thresholding has been done yet. As we will discuss later, thresholding will at some point also have noticeable effects on the relative error.

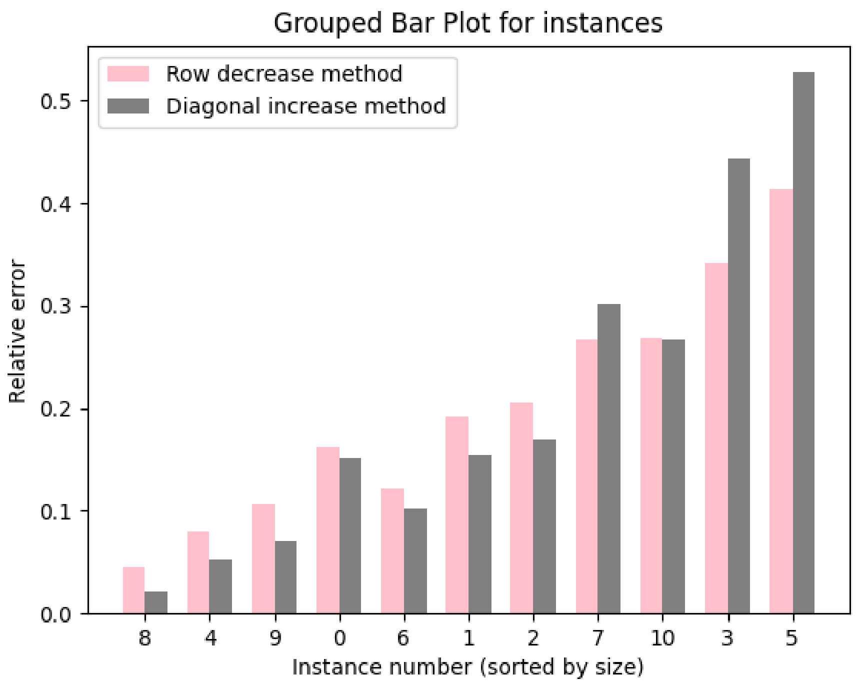

Figure 4: Relative error of diagonal dominance method for different instances

The two main takeaways from the graph are as follows. Firstly, the row decrease method (version 2) performs better on the larger instances whereas the diagonal increase method (version 1) performs better on smaller instances. Secondly, although not completely visible in this graph, the best-performing version and instance combination reached a relative error of at least 0.025.

The intuitive explanation for the first takeaway is: for smaller instances, the diagonal increase method performs better than the row increase method, since for small instances we are more likely to have to increase the diagonal relatively less than the rows. The opposite holds for large instances. There, it's more likely that we have to increase the diagonal a lot more relative to how much we have to decrease the rows.

Although the error mentioned in the second takeaway appears small, relative to the other instances, it is huge in comparison. Seeing as the relative error can be as large as 0.4 using the row decrease method (the best option for the largest instance, which in practice is still a relatively small instance), we must conclude the following: neither of the diagonal dominance methods can be relied upon when wanting to maintain a reasonable amount of information (low relative error) through the conversion. The method does however suit the situation in which a (variable) level of sparsity is valued higher than a low relative error. This is because of the fact that a strictly diagonally dominant matrix cannot be made non-diagonally dominant by removing off-diagonal values. Keeping this property in mind, as soon as we have obtained a strictly diagonally dominant matrix through either version of this method, we can threshold up to any threshold we please, where in the ultimate case we are left with a diagonal matrix. In the following graph, we see the influence on the relative error this thresholding of the strictly diagonally dominant matrix has.

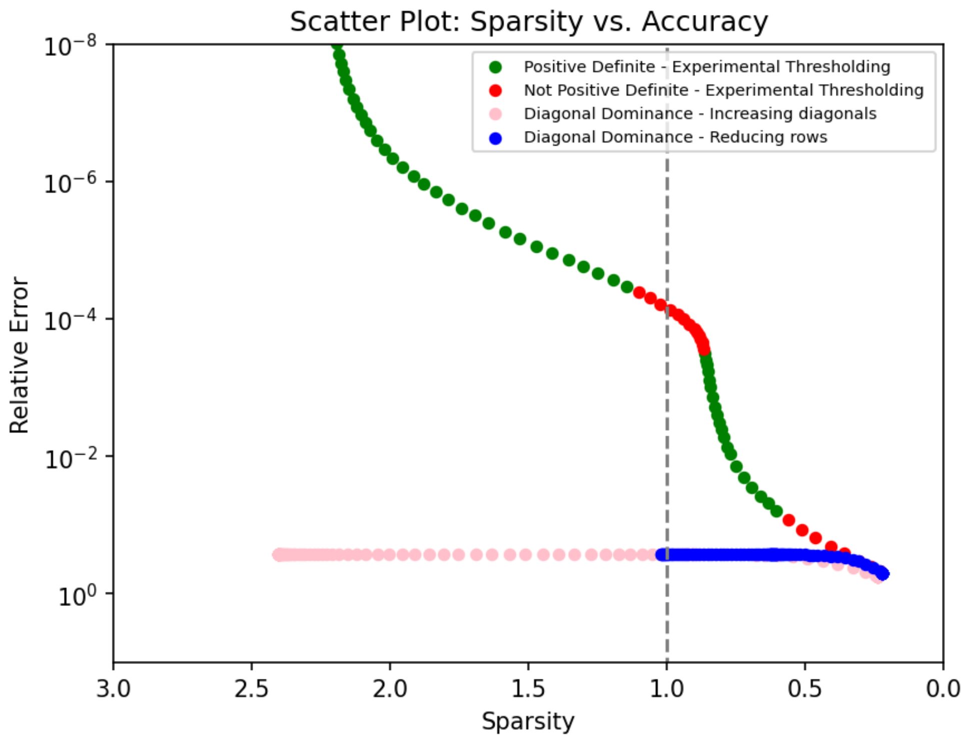

Figure 5: Performance of the diagonal dominance methods

5 Method: Lambda Perturbation

The initial solution tried by Sioux considered simple thresholding. This means that all values inside the matrix Σ_u which are below some threshold are erased. This allows one to remove all small covariances inside the matrix and therefore make it sparse without losing much of the original Σ_u. The only problem with this is the fact that simple thresholding does not preserve positive definiteness of the matrix which is required.

Hence, we would like to find a threshold that guarantees positive definiteness and also create enough sparsity. Furthermore, although experimental thresholding might work, we would like to find a theoretically sound threshold. We claim the following:

Lemma 5.1 (Positive definiteness of a perturbed matrix)**.** *Let A be a positive definite matrix and ΔA be a matrix such that $\hat{A} = A + \Delta A$. Then $\hat{A}$ is positive definite if*

$$\|\Delta A\|_2 < \|A^{-1}\|_2^{-1}.$$

Where $\|\cdot\|_2$ is the induced norm from the Euclidean metric.

Proof. The following proof is adapted from [2]. Let A be a positive definite matrix and ΔA be a matrix such that $\hat{A} = A + \Delta A$. Then, by definition of a postive definite matrix, we can write the eigenvalues of A as $\lambda_{\max} \geq \lambda_2 \geq \cdots \geq \lambda_{\min} > 0$. Moreover, we know that $\|A\|_2 = \lambda_{\max}$ and $\|A^{-1}\|_2^{-1} = \lambda_{\min}$. Furthermore, if $\{\eta_i\}_{1\leq i\leq n}$ are the eigenvalues of ΔA, we have that $\|\Delta A\|_2 = |\eta_j| = \max\{|\eta_i| \; |1 \leq i \leq n\}$. Then, by assumption, we have that

$$|\eta_j| = \|\Delta A\|_2 < \|A^{-1}\|_2^{-1} = \lambda_{\min}.$$

We argue by contradiction. Assume that $\hat{A}$ is not positive definite. Then, there exists a unit vector v such that

$$\begin{aligned}(\hat{A}v, v) &\leq 0\\ ((A + \Delta A)v, v) &\leq 0\\ (Av, v) + (\Delta Av, v) &\leq 0\\ (Av, v) &\leq -(\Delta Av, v).\end{aligned}$$

By checking the Rayleigh quotient and applying one of the appropiate minmax principles [3], we find that $(Av, v) \geq \lambda_{\min}$ and $(\Delta Av, v) \leq |\eta_j|$. Applying our inequality yields $\lambda_{\min} \leq |\eta_j|$ which is a contradiction to our assumption. Hence, $\hat{A}$ is positive definite. □

This identity will be the main ingredient of this method. Before we present our algorithm, we will make use of the following inequality [4]:

$$\|A\|_2 \leq \|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2}.$$

Where A is a matrix and $\|\cdot\|_F$ is the so-called Frobenius norm. Note that if for the matrix ΔA, we have that $\|\Delta A\|_F < \|A^{-1}\|_2^{-1}$, this inequality ensures that Theorem 5.1 holds.

The approach we take is the following. We remove elements from the original matrix Σ_u from smallest to lowest by adding the opposites to the matrix ΔA until the $\|\Delta A\|_F$ exceeds the smallest eigenvalue of Σ_u. We could also stop at an earlier point if necessary. The updated matrix is then given by $\Sigma_s = \Sigma_u + \Delta A$ which is ensured to be positive definite by Theorem 5.1. Pseudo-code of the algorithm can be found in Algorithm 4:

Algorithm 4 Turn Σ_u into a more sparse matrix Σ_s

```
 Input: covariance matrix $\Sigma_u$, threshold
 Output: positive definite matrix $\Sigma_s$
Compute $\lambda_{\min}$ from $\Sigma_u$
Initiate $\Delta A$ empty
while $\Sigma_u$ contains non-zero entries or smallest entry $>$ threshold do
    Find smallest entry that remains from $\Sigma_u$
    Add entry and its symmetric counterpart to $\Delta A$
    if $\|\Sigma_u + \Delta A\|_F \geq \lambda_{min}$ then
        Remove entry and its symmetric counterpart from $\Delta A$
        Break
    end if
end while
Return $\Sigma_u + \Delta A$
```

Before we present the results of this algorithm, we do a few more modifications to the algorithm. To begin, we found that there are a lot of values that are extremely small and therefore barely have an effect on the $\|\cdot\|_F$-norm compared to $\lambda_{\min}$. The loop procedure in our previous algorithm is quite expensive and therefore we would like to take some of these extremely small values out before we start the loop. Let n denote the number of non-zero entries in the updated covariance matrix Σ_u. Now, using Theorem 5.1, we yield that

$$\lambda_{\min} > \|\Delta A\|_F = \sqrt{\sum_{i,j} a_{ij}^2} \geq \sqrt{\sum_{i,j} \min_{k,\ell} a_{k\ell}^2} = \sqrt{n^2 \min_{k,\ell} a_{k\ell}^2}$$

From this we derive that

$$\begin{aligned} \lambda_{\min} &> \sqrt{n^2 \min_{k,\ell} a_{k\ell}^2} \\ \lambda_{\min}^2 &> n^2 \min_{k,\ell} a_{k\ell}^2 \\ \frac{|\lambda_{\min}|}{n} &> \min_{k,\ell} |a_{k\ell}| \end{aligned}$$

Now, we start by removing entries with absolute value smaller than this bound and then apply the method previously described. Note that applying this threshold changes the eigenvalues of Σ_u which means we must recompute the smallest eigenvalue between methods. A previous calculation proved to be erroneous which means that the results presented are relatively optimistic to what is achievable with this bound. However, note that the iterative approach after using this bound will compensate for this. Hence, the change in this bound mainly effects the running time and not the actual result.

Now, after we have applied this method to the matrix Σ_u, the new matrix Σ_s has new eigenvalues that might still be all larger than zero. Hence, we can apply this method iteratively to the output matrix. The result of this is given in purple in Figure 6. The purple lines moves from left to right as iterations increase (since we remove entries each iteration and hence increase sparsity). The total number of iterations is equal to 100. As we can see, the method approaches sparsity 1 from the left and follows the experimental thresholding trajectory. This makes sense since we are essentially thresholding with a theoretically sound value. From Figure 7, we find that the decrease in sparsity levels out as iteration increase. This makes sense since the values we remove become larger and larger whilst the smallest eigenvalue stays more or less the same.

The main advantage of this method over the other methods is the fact that it is theoretically completely sound. We are sure that we will maintain positive definiteness. The disadvantage of the method is the fact that it is quite expensive to execute. For each iteration the smallest eigenvalue must computed twice which is very expensive. We must compute the smallest eigenvalue for the initial thresholding which updates the matrix Σ_u, in turn updating the smallest eigenvalue of Σ_u. Hence, to ensure positive definiteness, we must again compute the smallest eigenvalue of Σ_u when starting iterations. Although the implementation might not be optimal, running 100 iterations on the initial dataset took about 20 minutes.

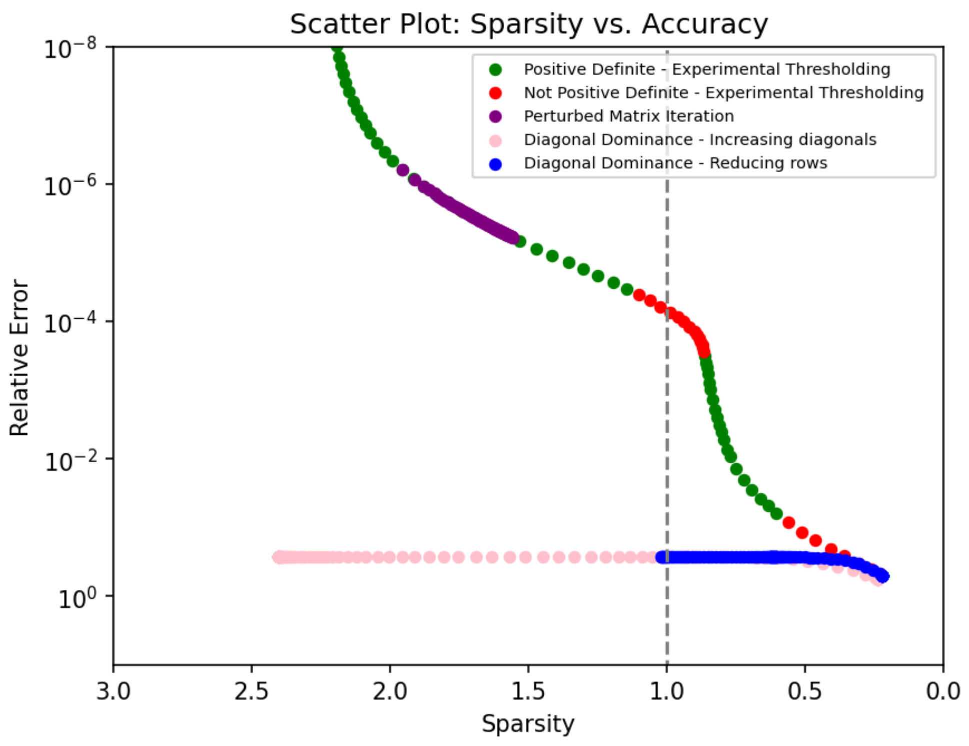

Figure 6: Performance of the iterative lambda perturbation method

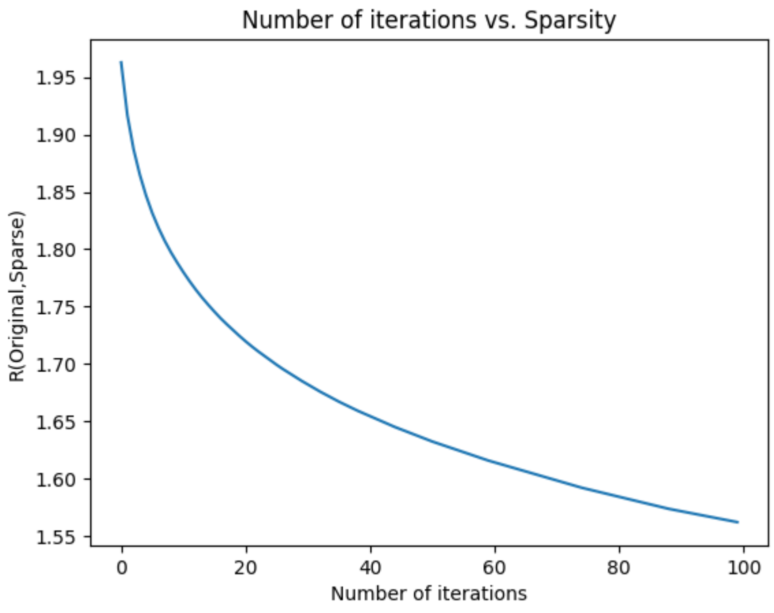

Figure 7: Sparsity compared to the number of iteration of the lambda perturbation method

6 Method: Symmetric matrix operations

Another way to ensure symmetry and positive definiteness of the sparsified matrix is by applying symmetric matrix operations. These go as follows: given a matrix A, we apply Gaussian elimination to the rows (say we add row j to row i), after which we immediately apply the same operation to the column with the same index (meaning we add column j to column i; analogous formulations for the other types of operations like scaling and interchanging rows). These actions are theoretically proven to preserve symmetry as well as positive definiteness (since the original matrix is congruent to the matrix produced after applying the symmetric matrix operations; see Sylvester's law of inertia [3]), which is why they are useful here.

However, when eliminating value (i, j) by adding a multiple of row j (since the diagonal element with indices (j, j) must always exist thanks to the matrix being positive definite) to row i, we run the risk of creating new nonzero values in row i, which thus does not guarantee an improvement in sparsity. This is why we will use symmetric matrix operations with a slight adaptation: when a value is 0 before the operation, it will remain 0. This means we will sometimes have to do "incomplete" operations, as described below in Algorithm 5:

Algorithm 5 Incomplete symmetric matrix operations

```
  Input: covariance matrix $\Sigma_u$, maximal error permitted $\varepsilon > 0$
  Output: sparser covariance matrix $\Sigma_s$
Let $S = \Sigma_u$
while $S$ positive definite and current error $< \varepsilon$ do
    Find minimum absolute value in $S$ with indices $(i, j)$
    Add $\frac{S_{ij}}{S_{jj}} \times$row $j$ to row $i$ to eliminate $S_{ij}$
    Add $\frac{S_{ij}}{S_{jj}} \times$column $j$ to column $i$ to eliminate $S_{ji}$
    Set new nonzero values back to 0
    Rescale row and column $i$ evenly such that diag$(S) =$ diag$(\Sigma_u)$
end while
Let $\Sigma_s = S\ \Sigma_s$
```

Of course, by altering the operations, the theory no longer ensures they preserve positive definiteness (symmetry is obviously still preserved). Although, thankfully, these incomplete symmetric matrix operations still preserve positive definiteness in many cases, as long as we do not delete values which are too big (which is ensured by keeping the maximal error permitted low). To ensure we never output a matrix which is no longer positive definite, we add a check every few iterations, after which we use a binary search over the iterations to efficiently find the most optimal, still positive definite matrix.

Lastly, note how we rescale the rows and columns such that the diagonal elements, the variances, stay untouched, as we intuitively do not want these to change.

As can be seen in Figure 8, this method closely follows the pattern described by the different variations on thresholding. This is to be expected, as symmetric matrix operations in sparse matrices frequently coincide with simple thresholding (or at least very close to it, by altering other values only very slightly). The relative error using the Frobenius norm is therefore always at least as big as that of thresholding, since symmetric matrix operations do still often alter more than just 1 value per iteration. However, similarly to thresholding, the method is likely to eventually break down, which it does (in this case) around the same point.

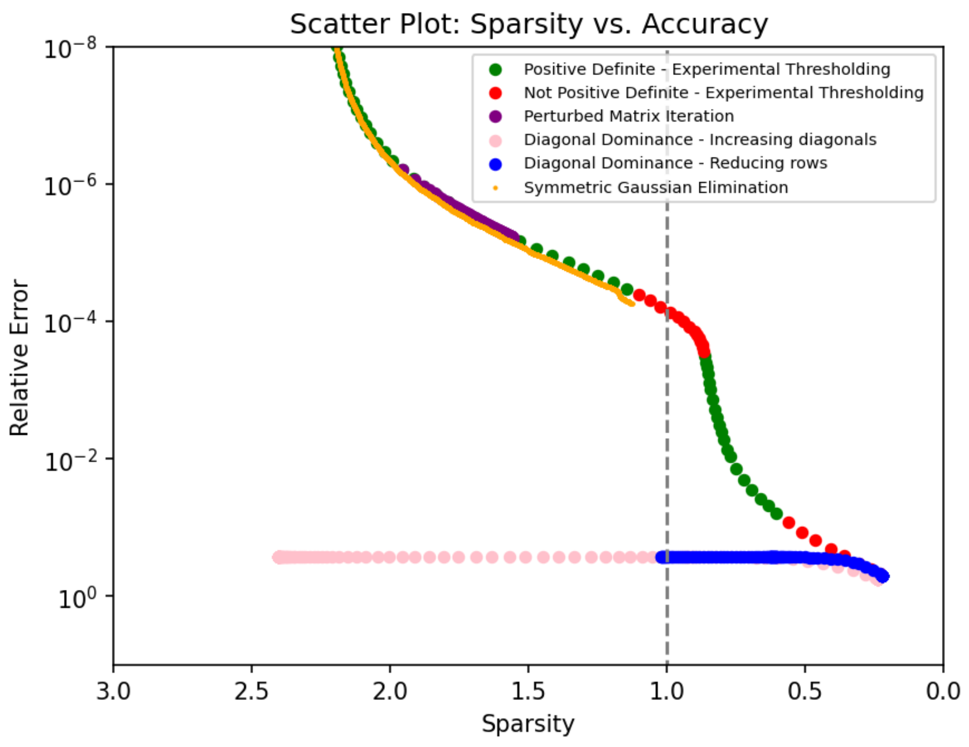

Figure 8: Performance of the symmetric matrix operations method

This method, though practically quite reliable, is relatively slow for very large, more dense matrices. Therefore, it is not advised to exclusively apply this method to try to sparsify the newly filled in covariance matrix. However, it does work well in further sparsifying a matrix on which we have already done some thresholding first. We will therefore first apply the method of lambda perturbations, and after apply the symmetric matrix operations until they break down. The result of this combination can be seen in Figure 9. While this does give a result slightly worse (in error to sparsity ratio) than had we solely applied symmetric matrix operations, it did do so very quickly, as the method often only has to do a few operations per iteration when the matrix is already quite sparse to begin with. Though further research should be done in order to verify this claim, at the very least in this particular example, it serves as a good and fast way to improve upon the results produced by the method of lambda perturbations.

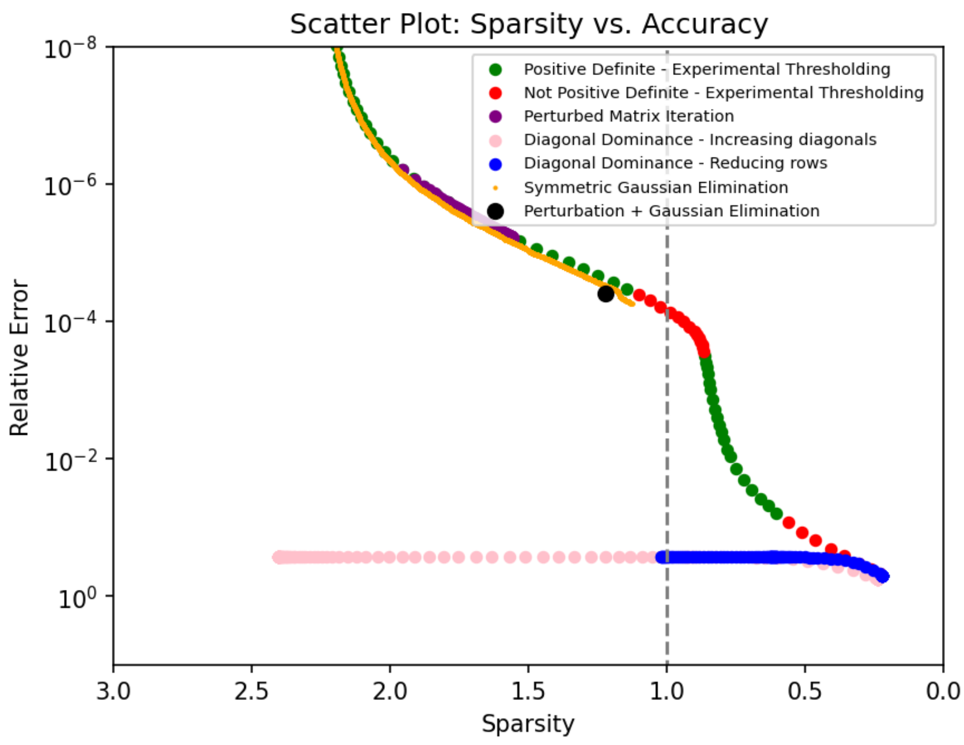

Figure 9: Performance of the symmetric matrix operations method after lambda perturbations

7 Conclusion

The goal of our project was to increase the sparsity of the covariance matrix, while ensuring positive definiteness of this matrix and minimize the relative error (so try to keep this matrix as close as possible to the original).
We found that both of our diagonal dominance methods have the highest relative error in comparison with the other methods, even though the diagonal dominance methods can deliver us a higher degree of sparsity at the cost of the accuracy.
The method of symmetric Gaussian elimination, where we perform only symmetric row operations, is a lot more accurate than the method just discussed. Although it isn't able to make the matrix as sparse, we do nearly reach a sparsity of 1. One downside is that the method isn't theoretically supported, meaning that we provide any theoretical guarantee for the performance of the method.
Using the Lambda Perturbation method exclusively, we obtain results that are along the same curve of the symmetric row operation method, but we can use the symmetric row operation method to get a little more sparsity at the cost of some more error. The upside of this method is that we have some theoretical guarantees linked to it, namely we are sure that we will maintain positive definiteness.
A combination of the Lambda Perturbation method and the symmetric row operation method upon investigation also lands us along the curve of the results of the symmetric row operation method.
According to us, the best method would be a combination of methods, specifically the symmetric row operation method in combination with the Lambda Perturbation method, since it

lands us along the curve of the symmetric row operation method, but we do get theoretical guarantees for our method, which is why we think this method has the best potential.

8 Future work

Regarding future work, it would be interesting to look at the following topics to explore in more detail:

- First of all, the metric must be determined that will be used to measure sparsity and accuracy of the matrix. The current measures used have advantage and disadvantages which should be considered carefully when choosing the final solution. The Frobenious norm considers all elements and therefore intuitively seems very useful to measure accuracy. However, since we consider all elements its computational complexity is quite high. One could also only look at the diagonal elements to measure accuracy.
- Furthermore, for the given methods, one should test them on more realistic and especially bigger problems. The current (implementations of) algorithms is far from optimal which means that running times might be fairly slow relative to the true potential of the solutions. A more refined version of the methods presented might offer a good solution for the problem in question.
- Next, the big theoretical question remains is why the experimental thresholding shows an alternating pattern when it comes to positive definiteness of the matrix. If one figures out why this is the case, they might be able to take advantage of the reason to develop an efficient method.
- Fourthly, since the the variances within the covariance matrix can be any positive real number, it might be more suitable to instead use the scaled correlation matrix. The correlation matrix forces the diagonal elements to be scaled to 1, as well as the off-diagonal elements to be within the interval $[-1, 1]$ instead of any real number. This effect should be further investigated upon, though we will not touch upon this here.
- Penultimately, a suggestion was made that combines two of our current methods. This suggestion entails that we first apply thresholding to the matrix untill we have a desired number of zeros and then apply diagonal dominance by increasing the diagonal. The problem that could arise is the fact that we might lose a lot of accuracy when increasing this diagonal.
- Finally, a suggestion was made to look at row reductions and Gaussian elimination in more detail.

References

[1] J. Franken, W. Florijn, M. Hoekstra, and E. Hagemans, “Rebuilding the cadastral map of the netherlands: The artificial intelligence solution,” *Proceedings of [Conference Name]*, 2020. Available at: `https://fig.net/resources/proceedings/fig_proceedings/fig2020/papers/ts08h/TS08H_franken_florijn_et_al_10523.pdf`.

[2] D. T. (https://math.stackexchange.com/users/13152/dustin tran), “Preservation of positive-definiteness from small perturbations.” Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/527605 (version: 2013-10-15).

[3] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, UK: Cambridge University Press, 1990.

[4] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD, USA: Johns Hopkins University Press, 3rd ed., 1996.

A Thresholding

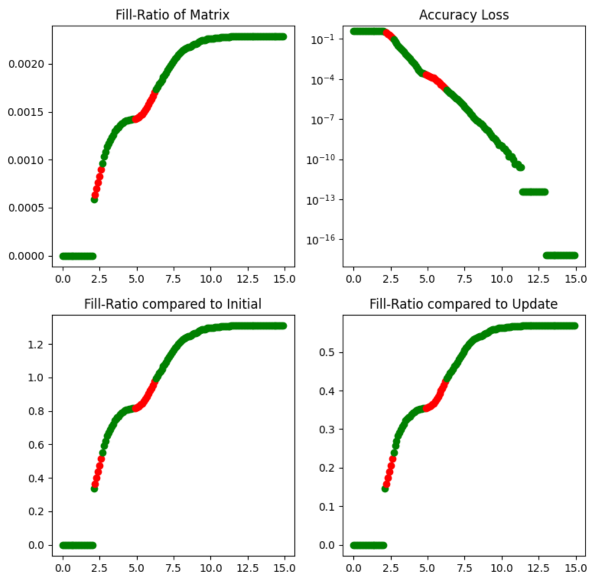

Figure 10: Thresholding for Data Set 0

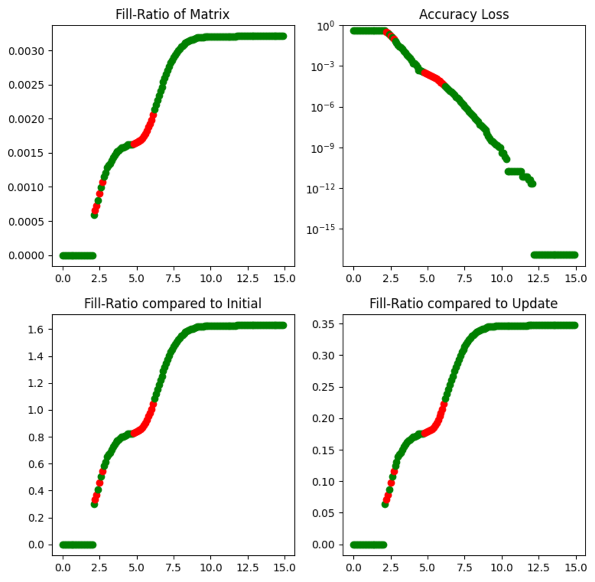

Figure 11: Thresholding for Data Set 1

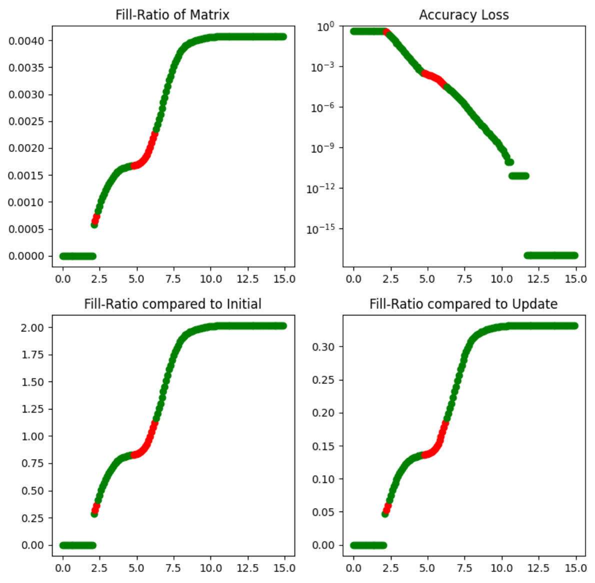

Figure 12: Thresholding for Data Set 2

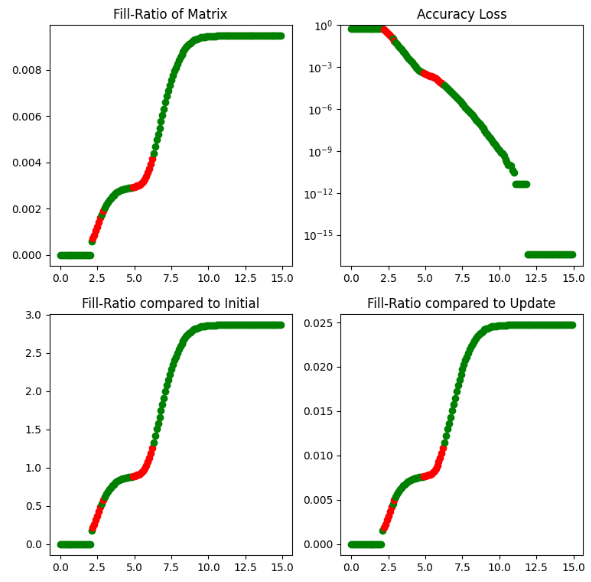

Figure 13: Thresholding for Data Set 3

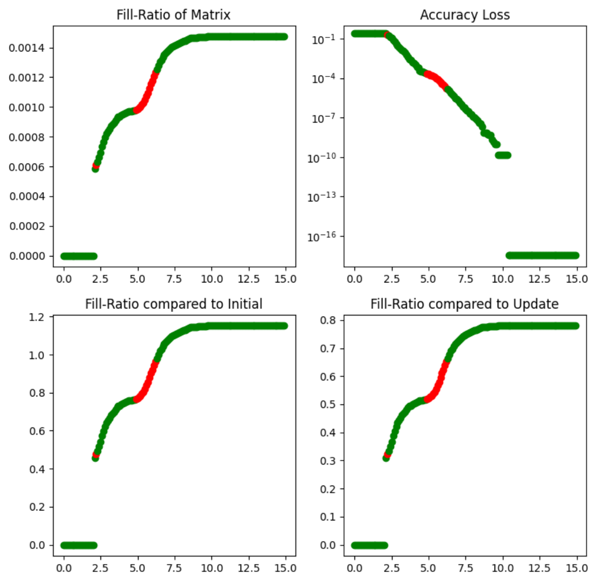

Figure 14: Thresholding for Data Set 4

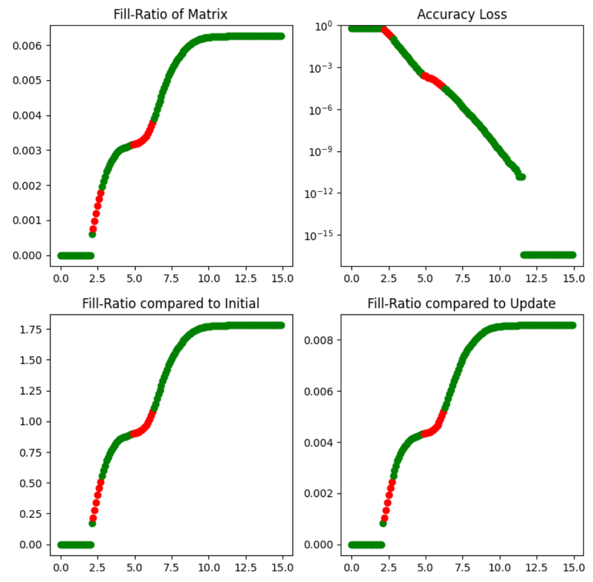

Figure 15: Thresholding for Data Set 5

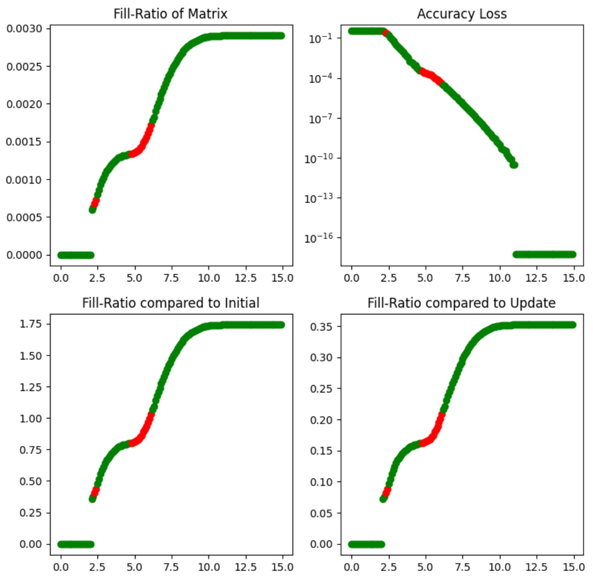

Figure 16: Thresholding for Data Set 6

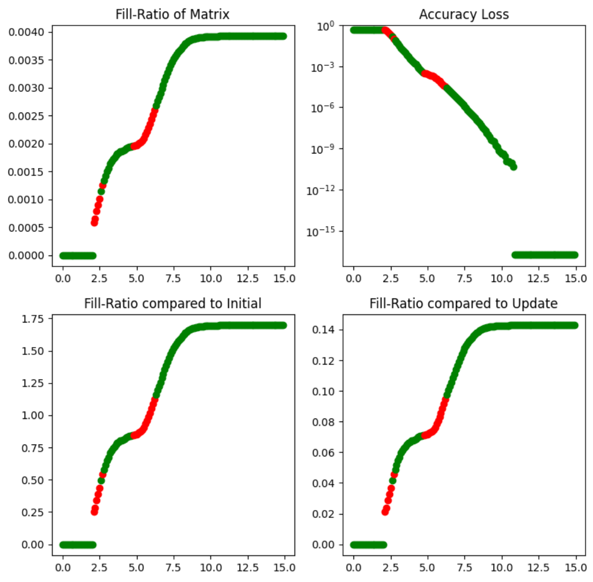

Figure 17: Thresholding for Data Set 7

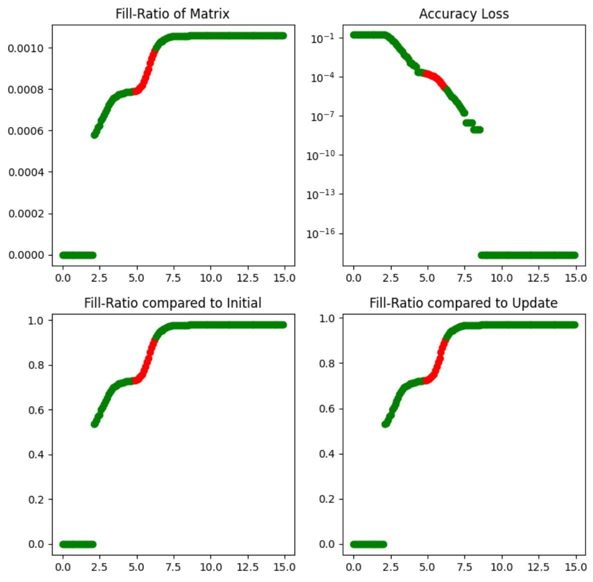

Figure 18: Thresholding for Data Set 8

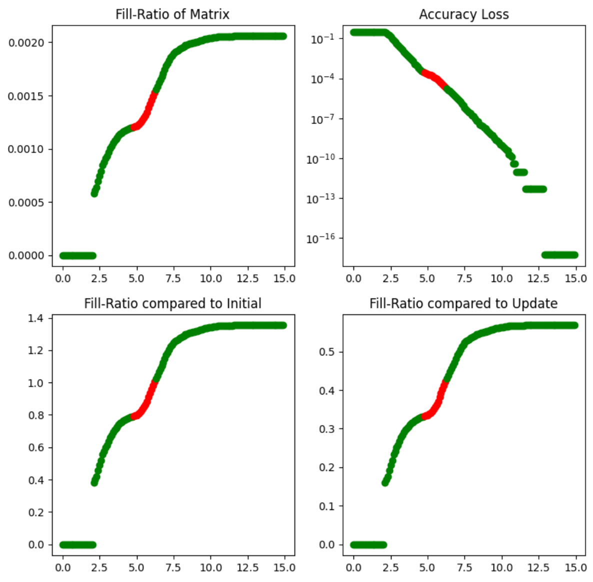

Figure 19: Thresholding for Data Set 9