POLITECNICO MILANO 1863 | SCHOOL OF CIVIL, ENVIRONMENTAL AND LAND MANAGEMENT ENGINEERING

Efficient Leave-One-Block-Out Outlier Detection

MASTER OF SCIENCE DEGREE IN

GEOINFORMATICS ENGINEERING

Author: **Jianwei Deng**

Lesson: Geoinformatics Project

Student ID: 10973015 / 245188

Advisor:Ludovico Giorgio Aldo Biagi

Co-Advisor:Nikolina Zallemi

Official Professor:Mariagrazia Fugini

Academic Year: 2025/7/30

Abstract

Outlier detection is of great significance in fields such as geodesy, spatial analysis, and regression modeling. In practice, the presence of outliers or corrupted data in observational data is a common problem, which can significantly affect the accuracy of parameter estimation. The traditional search method is the leave-one-block-out (LOBO), which focuses on iteratively removing certain data blocks from a dataset containing n observation samples for training.However, when the data is randomly divided and the sample size is large, this method faces issues such as limited sensitivity and low operational efficiency.

To solve the above problems, we propose a novel and efficient outlier detection tool based on LOBO, the efficient leave-one-block-out (ELOBO) method. The main innovation of this tool is to use the Q matrix representing the observation correlation to automatically extract highly correlated data blocks through the depth-first search (DFS) algorithm and adopt a new efficient calculation formula. This improved mathematical method avoids the repetition of model iteration and can use each sample as test data to quickly derive prediction errors and performance indicators under a limited number of global model calculations. Compared with the traditional LOBO that requires explicit fitting of n models, ELOBO significantly improves the computational efficiency and result estimation speed.

At the same time, considering the tool’s intuitiveness and user experience, we developed an interactive web application platform based on Streamlit. The platform allows users to upload data files online, freely choose different types of Q matrix structures, perform parameter estimation through the least squares method, quickly detect abnormal observations, and clearly display residual results and block structures through a graphical interface. At the same time, the platform provides exportable CSV format data results to further analysis by the user. This study draws on existing relevant literature to optimize the characteristics of fast calculation and estimation, building on the traditional LOBO method.

Key-words: Leave-One-Block-Out(ELOBO); Covariance Matrix Q; Depth-First Search (DFS); Residual Analysis; Least Squares Estimation; Streamlit Application;

Contents

1 Chapter one

In this chapter all useful information about the project are reported.

1.1. Introduction

1.1.1. Background and Motivation

Nowadays, in applications of geodetic networks such as environmental monitoring and sensor arrays, observations may be systematically grouped or correlated due to design structures or spatial layouts. This is why traditional point-by-point outlier detection methods cannot meet existing needs.

I initially started this work because I wanted to explore how to identify anomalous subsets of observations that are correlated. More importantly, if the computational cost of full revaluation is so high, we can imagine whether there is an efficient way to solve this problem?

To solve this problem, we adopted an advanced leave-one-out (ELOBO) method. Our technique reveals a systematic modeling process, using a graph-based method (using the DFS traversal principle) to design the covariance matrix Q and modeling with its help, while using advanced matrix reuse techniques, including reuse of large matrix results + small matrix fast processing, which greatly reduces the computational cost and improves the operation efficiency while avoiding repeated fitting and large-scale matrix operations.

1.1.2. Objective of the Project

This project shows an interactive tool made by Python. It uses Efficient Leave-One-Block-Out (ELOBO) method to detect outliers. The app is built with Streamlit. It is for checking block structure in observation data, so it can do statistical estimation and outlier detection in a strong way.

The tool helps user to:

· Estimate parameters from observation data.

· Find outlier blocks by covariance matrix Q.

· Visualize residuals and correlation structure.

· Export table results for later use.

The system lets user interact flexibly. User can upload matrix (A, b, y, Q), see visualization, get blocks by DFS, and do fast linear algebra for ELOBO. This avoids fitting again and makes calculation faster.

1.2. Problem Formulation

1.2.1. Task Definition

The main task of the project is to make an interactive and fast tool to detect outliers in block-correlated data by efficient leave-one-out (ELOBO) method. User uploads system of equations with matrix A, b, y, gives covariance matrix Q, then tool can estimate parameters, find outlier blocks fast, and show residuals.

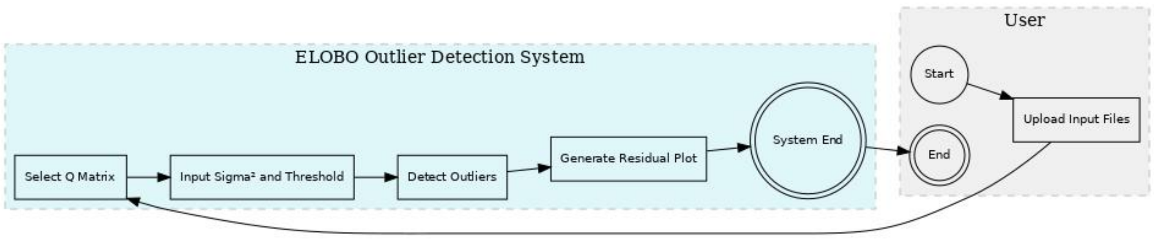


Figure : User Case Diagram

Its meaning is like this:

The User on the right means the user of the system. The oval part on the right means the boundary of the system. Its name is "ELOBO Outlier Detection System".

Goals:

· User uploads data (A, b, y) and Q structure settings

· A (m × n): design matrix

· b (m × 1): offset vector

· y (m × 1): observation values

· Let user define weight matrix Q flexible, with four types: identity, diagonal, block-diagonal, full matrix

· Do least squares estimation with general covariance structure

· Get block structures from Q by graph method (DFS)

· Do Leave-One-Block-Out analysis fast for each block

· Find blocks that are outliers by change of residual norm

· Show residual plots, outlier blocks found, and let user export results

1.2.2. Challenges

It is clear that the ELOBO framework is already in place. Yet, there are still some implementation problems in this regard, which might be called practical and potential issues:

· Handling Correlated Observations via Q Matrix

Another issue is that of handling the structured covariance matrices (Q). Under standard conditions of independent observations and Q full of blocks, populations of observations correlate within themselves, rendering the results as follows:

·Matrix inversion operations

·Residual computation (when blocks are correlated)

· Design a general method to accept four types of Q inputs (unit, diagonal, block diagonal, full)

· Leave-One-Block-Out (ELOBO) Implementation

The implementation of ELOBO entails the re-transmission of LS for every block left out, which if efficiently implemented must ensure:

·Avoid full matrix inversions by sequential

·application Reuse of already evaluated specialists.

·Ensure numerical stability on matrix partitioning and inversion

· Detecting Blocks from Arbitrary Q

To pull out blocks from arbitrary Q matrices, a graph-invoked approach is to be made. This is incorporated by:

Formulating matrices as adjacency graphs.

DFS procedure to determine connected components.

Possessing the power to isolate nodes or to eliminate any invalid sub-matric

· Stream-lit interface complexity

All functions for engagement/interface include:

Dynamically change the UI to fit the chosen Q format.

Manage session state during the prediction and recognition phase.

Provide badly formed input error messages for the incorrect size.

1.3 Related Work:LOBO Research

Outlier detection in Least Squares (LS) adjustment is long time a basic problem in regression and geodetic modeling. Classical LS thinks all observations are correct, but in real data there are big errors or outliers. LS gives best estimates if data is Gaussian, but it is not against outliers. One outlier can change the whole solution and make it hard to find and check.

To solve this, many diagnostic methods were made:

· Residual-based tests (like standardized residuals, t-tests) can check outliers, but they work worse when observations have correlation.

· Influence measures (like Cook’s Distance or Mahalanobis Distance) help find leverage points, but still think observations are independent.

· Leave-One-Out (LOO) gives better robustness because it looks at removing one observation at a time, but it does not see group correlation.

1.3.1 ELOBO as a Robust Alternative

The Leave-One-Block-Out (LOBO) method solves these limits by removing whole blocks of correlated data from the model. If the data vector is split into uncorrelated small vectors (blocks), leaving out one block and checking its effect makes the model more strong.

But normal LOBO has high compute cost because it needs to do full LS for each block. To fix this, researchers gave efficient ELOBO method:

· Do full LS only one time.

· Use algebra way to get block influence, no need to do full compute again.

· Can get residual and variance of each block with very small compute.

Two examples in papers of “An Efficient Leave One Block Out approach to identify outliers”shows:

· ELOBO is more reliable than normal LS residual tests.

·It has high number efficiency so can work on big data.

1.3.2.Methods Features

While old papers focus on theory and math speed of ELOBO, they do not give easy tools with good interface. Our project builds on that and does:

· Make a visual and interactive Python tool for ELOBO.

· Support flexible Q input (identity, diagonal, block-diagonal, full).

· Auto find correlation blocks by graph DFS.

· Show results by heatmap, residual plot, outlier table.

Different from old work, our system fills the gap between theory and real use. It gives researchers an easy and fast way to do block outlier detection on correlated data.

1.4. Proposed Method

1.4.1 Overview Methodology

We propose an efficient block-based outlier detection method, called ELOBO (Efficient Leave-One-Block-Out). This method accounts for the correlation between observations by using a covariance matrix Q. Unlike traditional leave-one-out methods, ELOBO primarily removes entire blocks of correlated observations and evaluates their impact on the residual structure of the model.

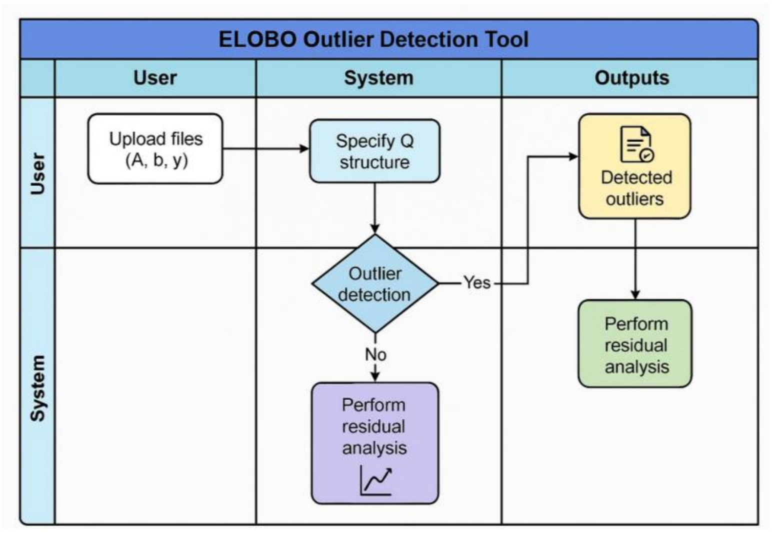


Figure : Lane_Diagram of Methodology

1.4.2 Methodology Details

Step 1: Q Matrix Analysis and Block Formation

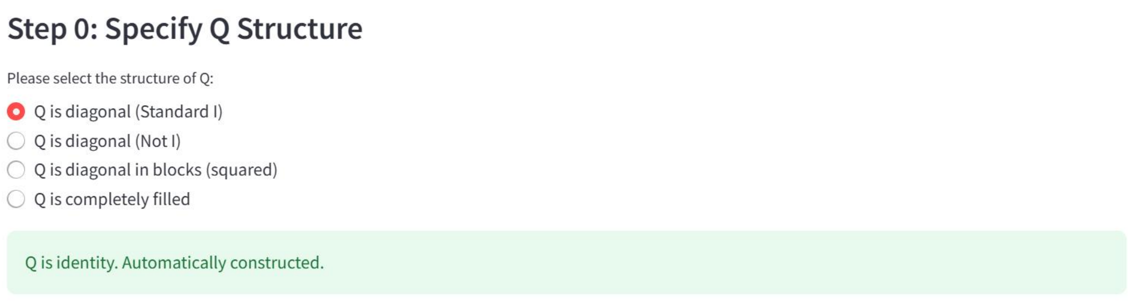
Type 1: Q is diagonal (Standard I)

·Description: All observations are independent and equally weighted.

· Processing: Q is set as an identity matrix Q=I, no correlation between observations.

·Block Formation: Each observation is treated as an individual block.

(Eg:Block = [[0], [1], ..., [m-1]])



Step 0: Specify Q Structure

Please select the structure of Q:

[✓ Q is diagonal (Standard I)]
Q is diagonal (Not I)
Q is diagonal in blocks (squared)
Q is completely filled

Q is identity. Automatically constructed.

Figure: Specify Q Structures 1

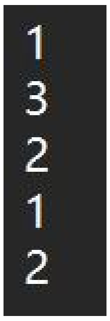
Type 2: Q is diagonal (Not I)

Description: Observations are still independent, but have different weights (variances).

Processing: Q is a diagonal matrix with custom diagonal entries.

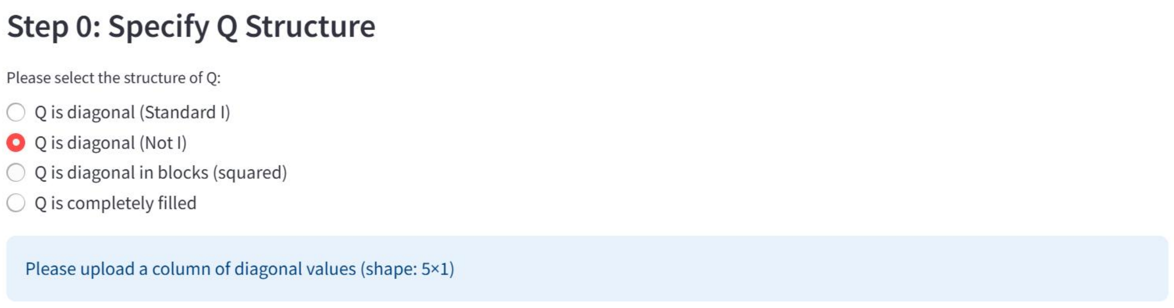
Block Formation: Same as Type 1 — each observation is its own block.

Block = [[0], [1], ..., [m-1]]



```
1
3
2
1
2
```

Figure：Data inputs for type 2



Step 0: Specify Q Structure

Please select the structure of Q:

- Q is diagonal (Standard I)
- [✓ Q is diagonal (Not I)]
- Q is diagonal in blocks (squared)
- Q is completely filled

Please upload a column of diagonal values (shape: 5×1)

Figure: Specify Q Structure

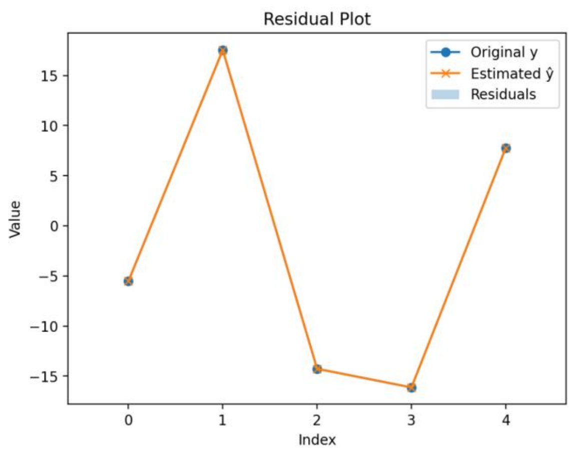
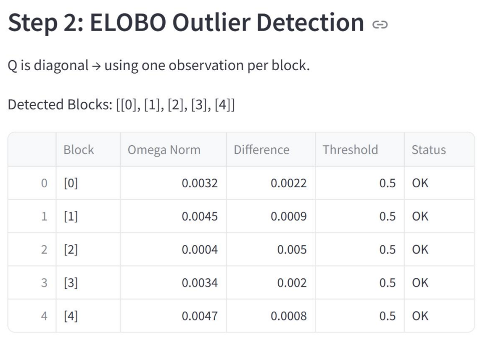


Figure: Residual Plot (Step1)



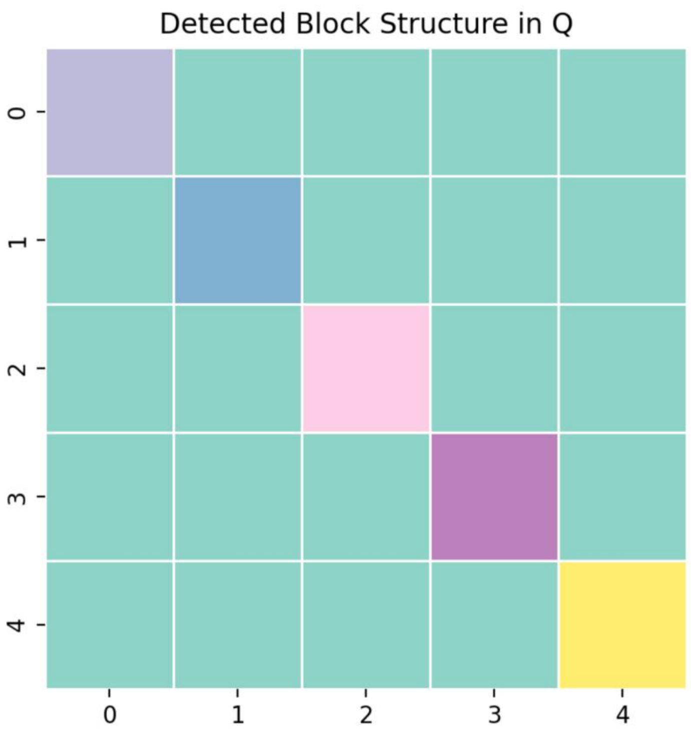
Step 2: ELOBO Outlier Detection

Q is diagonal → using one observation per block.

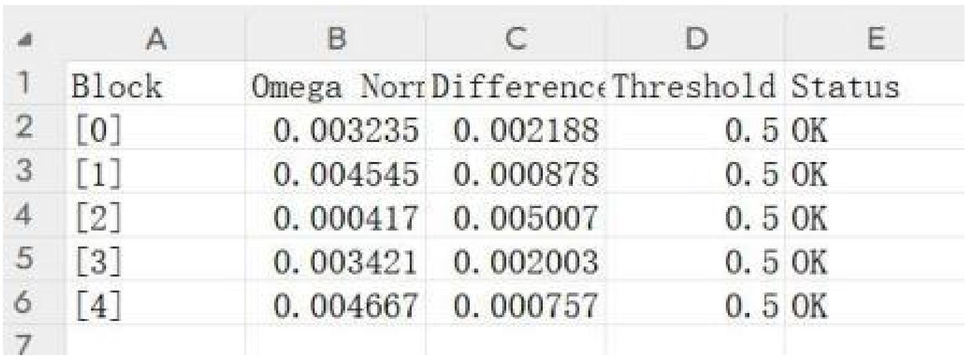
Detected Blocks: [[0], [1], [2], [3], [4]]

	Block	Omega Norm	Difference	Threshold	Status
0	[0]	0.0032	0.0022	0.5	OK
1	[1]	0.0045	0.0009	0.5	OK
2	[2]	0.0004	0.005	0.5	OK
3	[3]	0.0034	0.002	0.5	OK
4	[4]	0.0047	0.0008	0.5	OK

Figure: Output Of Outlier Detection(Step2)



Figure： Detected Block Structure in Q(Step2)



	A	B	C	D	E
1	Block	Omega Norr	Difference	Threshold	Status
2	[0]	0.003235	0.002188	0.5	OK
3	[1]	0.004545	0.000878	0.5	OK
4	[2]	0.000417	0.005007	0.5	OK
5	[3]	0.003421	0.002003	0.5	OK
6	[4]	0.004667	0.000757	0.5	OK
7					

Figure： Output Block to csv.file(Step2)

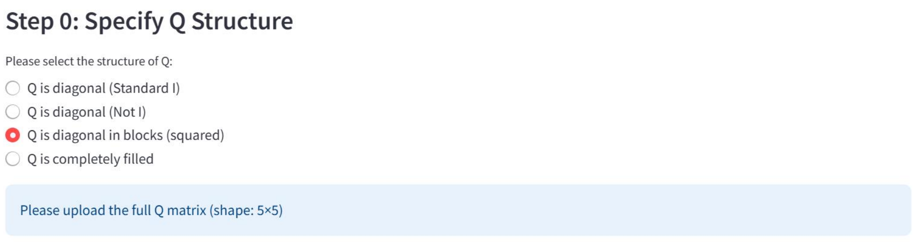
Type 3: Q is diagonal in blocks (squared)

Description: Observations are grouped and correlated within blocks; block structure is visible in Q.

Processing: Perform Depth-First Search (DFS) on Q's non-zero off-diagonal entries to extract connected subgraphs (blocks).

Block Formation: Each connected group of correlated observations forms a block.

Example: [[0, 1], [2, 3, 4]]



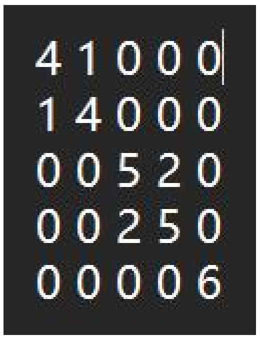
Step 0: Specify Q Structure

Please select the structure of Q:

- Q is diagonal (Standard I)
- Q is diagonal (Not I)
- [✓ Q is diagonal in blocks (squared)]
- Q is completely filled

Please upload the full Q matrix (shape: 5×5)

Figure: Specify Q Structures 3



```
4 1 0 0 0
1 4 0 0 0
0 0 5 2 0
0 0 2 5 0
0 0 0 0 6
```

Figure：Data inputs for type 3

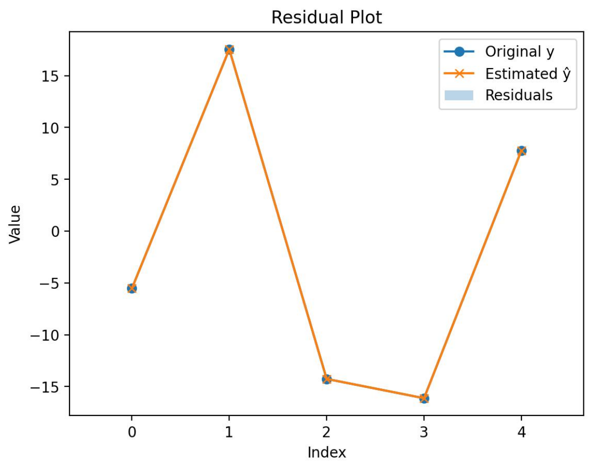
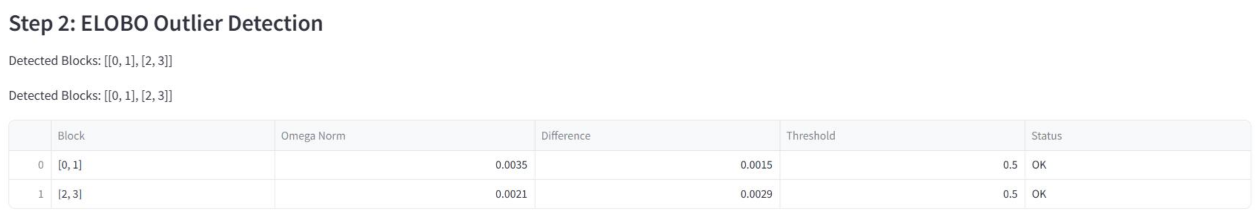


Figure: Residual Plot (Step1)



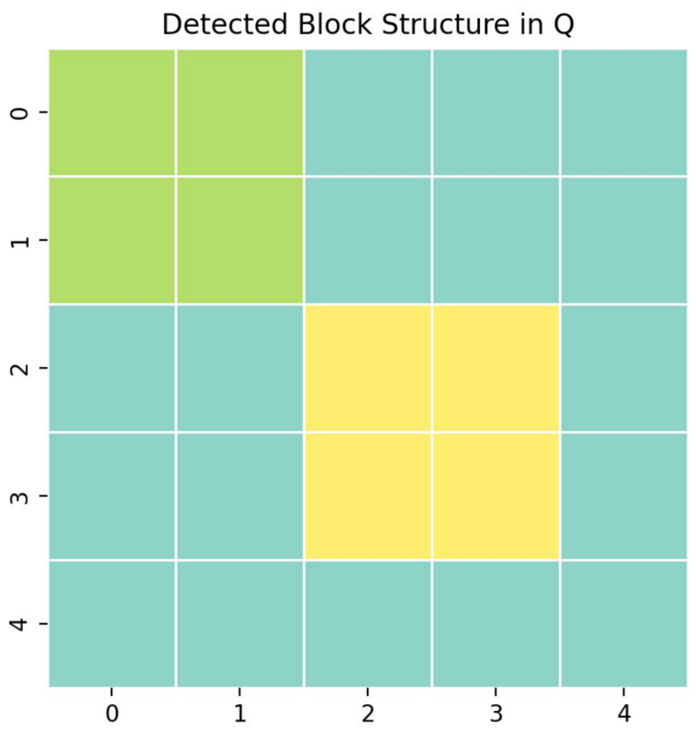
Step 2: ELOBO Outlier Detection

Detected Blocks: [[0, 1], [2, 3]]

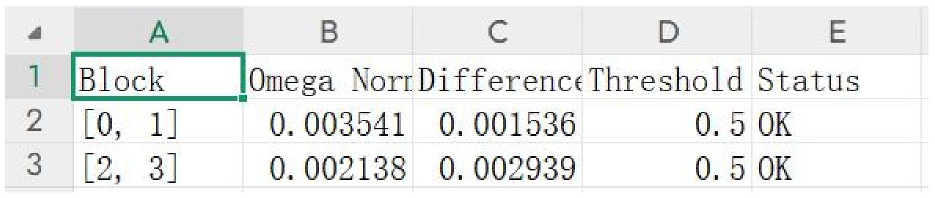
Detected Blocks: [[0, 1], [2, 3]]

	Block	Omega Norm	Difference	Threshold	Status
0	[0, 1]	0.0035	0.0015	0.5	OK
1	[2, 3]	0.0021	0.0029	0.5	OK

Figure： Output Of Outlier Detection(Step2)



Figure： Detected Block Structure in Q(Step2)



	A	B	C	D	E
1	Block	Omega Norr	Difference	Threshold	Status
2	[0, 1]	0.003541	0.001536	0.5	OK
3	[2, 3]	0.002138	0.002939	0.5	OK

Figure： Output Block to csv.file(Step2)

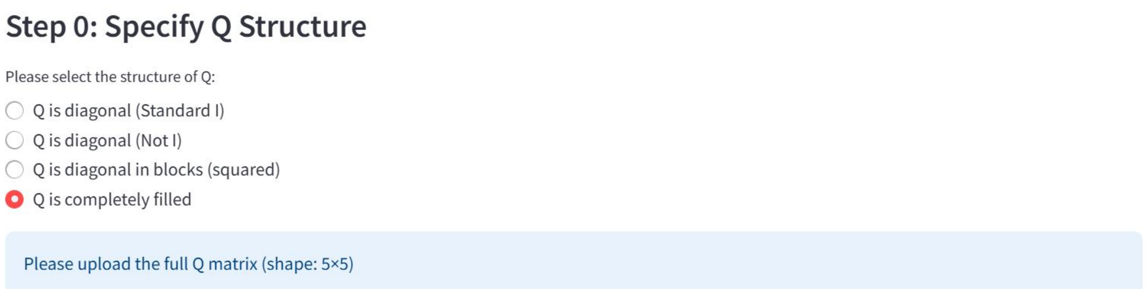
Type 4: Q is completely filled

Description: All observations are correlated with each other; Q is fully populated.

Processing: Treated as a single group; only one block exists.

Block Formation: All observations in one block.

Block = [[0, 1, ..., m-1]]

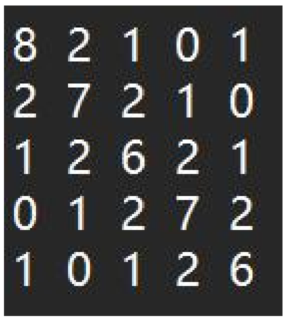

Step 0: Specify Q Structure

Please select the structure of Q:

- ○ Q is diagonal (Standard I)
- ○ Q is diagonal (Not I)
- ○ Q is diagonal in blocks (squared)
- [✓ Q is completely filled]

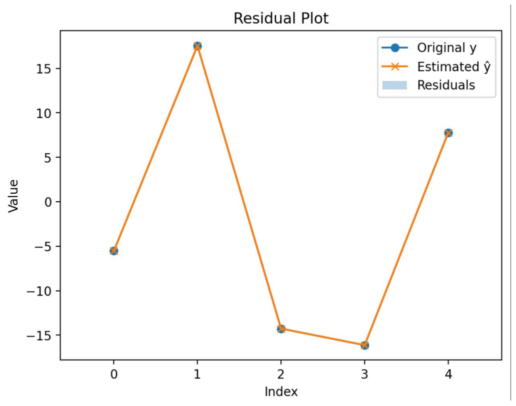
Please upload the full Q matrix (shape: 5×5)

Figure： Specify Q Structure

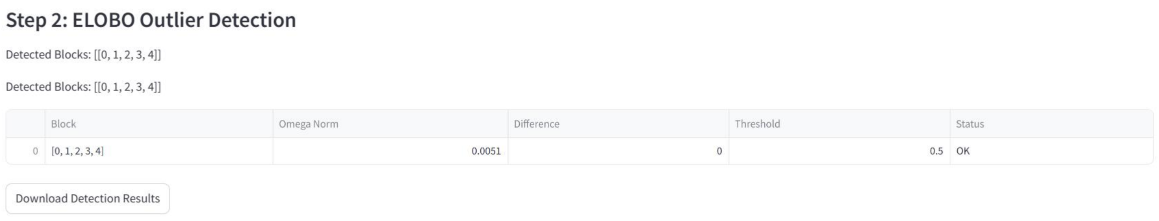


8	2	1	0	1
2	7	2	1	0
1	2	6	2	1
0	1	2	7	2
1	0	1	2	6

Figure：Data inputs for type 4



Figure： Residual Plot (Step1)



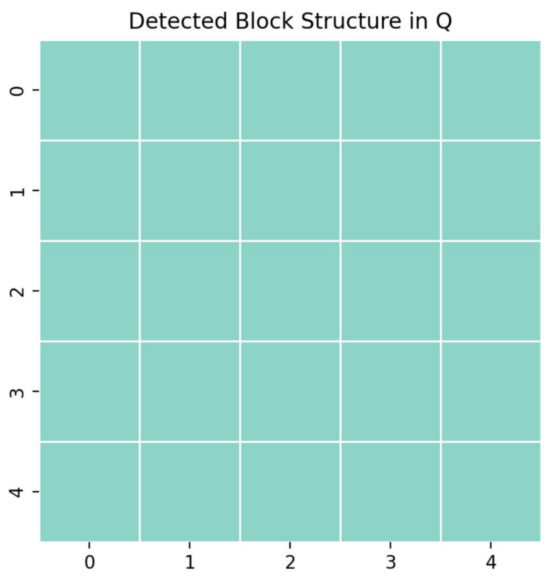
Step 2: ELOBO Outlier Detection

Detected Blocks: [[0, 1, 2, 3, 4]]

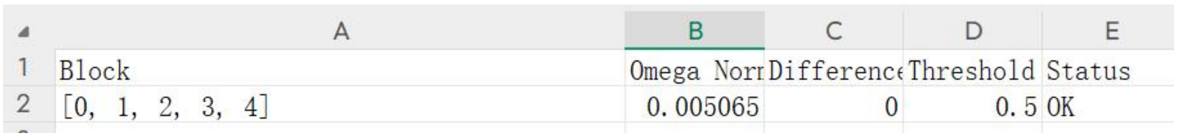
Detected Blocks: [[0, 1, 2, 3, 4]]

	Block	Omega Norm	Difference	Threshold	Status
0	[0, 1, 2, 3, 4]	0.0051	0	0.5	OK

Figure： Output Of Outlier Detection(Step2)



Figure： Detected Block Structure in Q(Step2)



	A	B	C	D	E
1	Block	Omega Norr	Difference	Threshold	Status
2	[0, 1, 2, 3, 4]	0.005065	0	0.5	OK

Figure： Output Block to csv.file(Step2)

Step 2: Least Squares Estimation

Given matrix inputs A, b, and observation vector y, the system solves the regularized normal equation:

$$\hat{x} = (A^T Q^{-1} A)^{-1} A^T Q^{-1} (y - b)$$

This produces estimated parameters,fitted values,and residuals The residuals serve as a baseline for comparison in subsequent ELOBO steps.

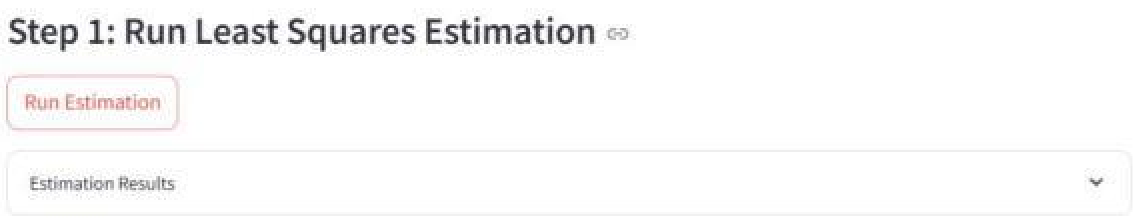


Figure: Run LS Estimation

Step 3: Efficient Leave-One-Block-Out (ELOBO)

Instead of re-solving the least squares problem from scratch for each block, the algorithm efficiently reuses the inverse of the normal matrix , along with matrix algebra, to evaluate the effect of removing each block. For block i, the method computes:

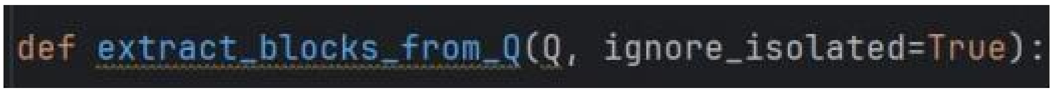
The difference in residual norm when block i is removed.

An influence measure (omega norm) derived from the adjusted residual and block covariance.A comparison to user-defined threshold to flag whether the block is an outlier.

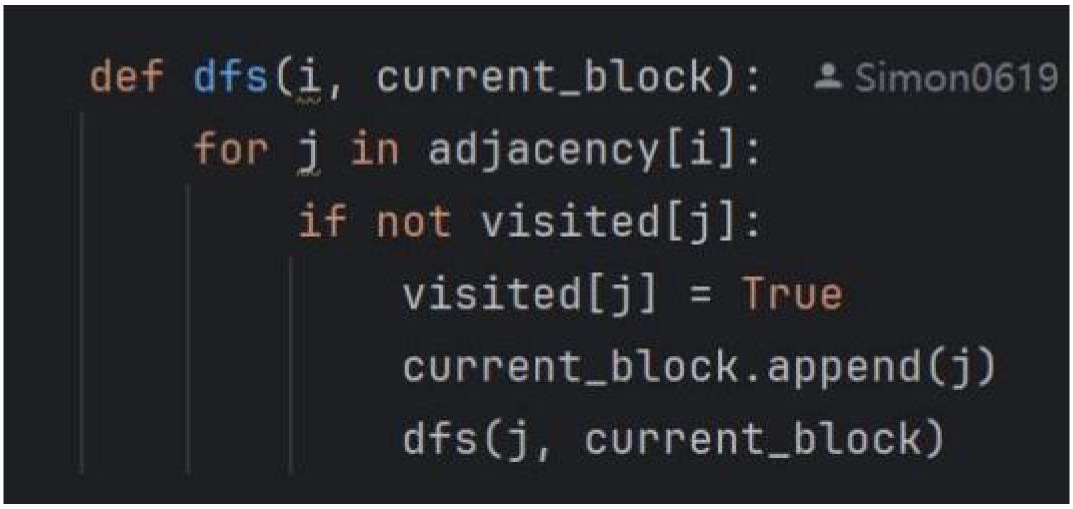
This design not only avoids redundant calculations but also significantly improves runtime, especially in the processing of high-dimensional problems.

·Block Structure Extraction:

Using DFS to traverse the adjacency graph (the connectivity defined by the non-zero elements in Q) and identify a connected block.



```python
def extract_blocks_from_Q(Q, ignore_isolated=True):
```



```python
def dfs(i, current_block):
    for j in adjacency[i]:
        if not visited[j]:
            visited[j] = True
            current_block.append(j)
            dfs(j, current_block)
```

Figure: DFS Block Extraction

•ELOBO Efficient Calculation:

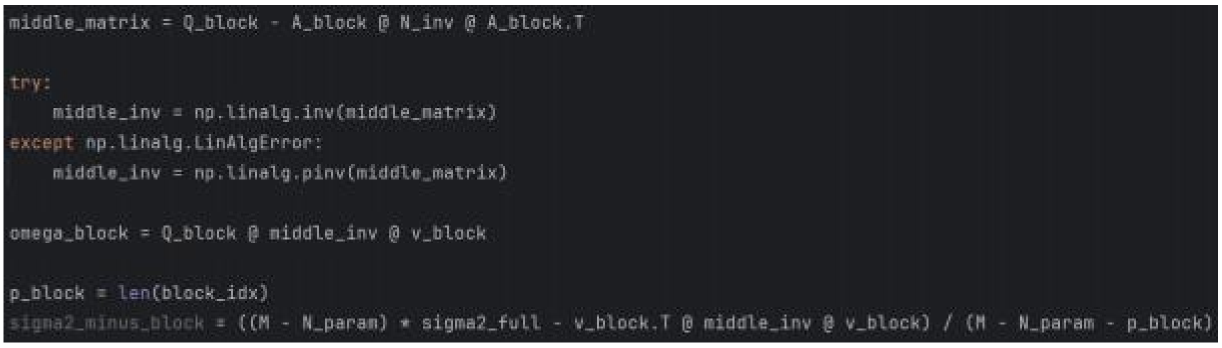
This includes calculating the change in the estimated vector, sigma² after removing block k, and the omega vector (detection core) as a detection indicator through norm(omega_block) and comparing it with the residual norm to determine whether the block is abnormal.

$$\hat{\mathbf{x}}_{(-k)} = \hat{\mathbf{x}} - \mathbf{N}^{-1}\mathbf{A}_k^T \left(\mathbf{Q}_k - \mathbf{A}_k\mathbf{N}^{-1}\mathbf{A}_k^T\right)^{-1} \hat{\mathbf{v}}_k \quad (4)$$

$$\sigma^2_{(-k)} = \frac{\left[(M-N)\hat{\sigma}^2 - \hat{\mathbf{v}}_k^T \left(\mathbf{Q}_k - \mathbf{A}_k\mathbf{N}^{-1}\mathbf{A}_k^T\right)^{-1} \hat{\mathbf{v}}_k\right]}{M-N-p_k} \quad (5)$$

$$\omega_k = \mathbf{Q}_k \left(\mathbf{Q}_k - \mathbf{A}_k\mathbf{N}^{-1}\mathbf{A}_k^T\right)^{-1} \hat{\mathbf{v}}_k \quad (6)$$

Figure: Reference Formula for ELOBO



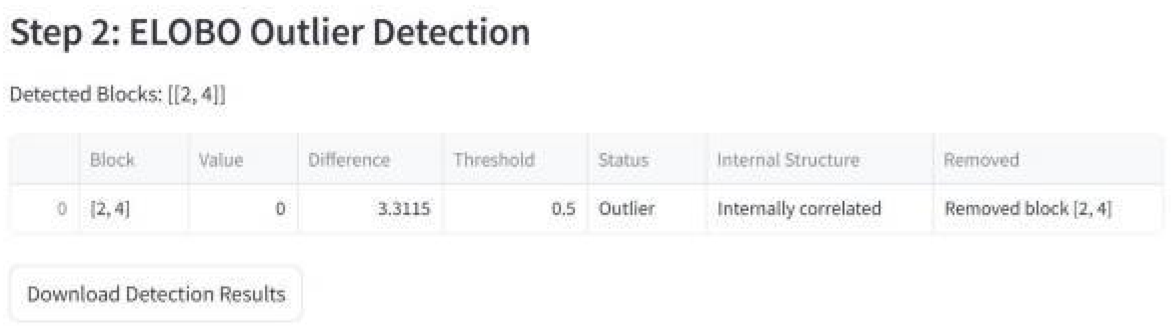
```
middle_matrix = Q_block - A_block @ N_inv @ A_block.T

try:
    middle_inv = np.linalg.inv(middle_matrix)
except np.linalg.LinAlgError:
    middle_inv = np.linalg.pinv(middle_matrix)

omega_block = Q_block @ middle_inv @ v_block

p_block = len(block_idx)
sigma2_minus_block = ((M - N_param) * sigma2_full - v_block.T @ middle_inv @ v_block) / (M - N_param - p_block)
```

Figure: ELOBO Efficient Calculation



Step 2: ELOBO Outlier Detection

Detected Blocks: [[2, 4]]

	Block	Value	Difference	Threshold	Status	Internal Structure	Removed
0	[2, 4]	0	3.3115	0.5	Outlier	Internally correlated	Removed block [2, 4]

Figure: Threshold Setup& ELOBO Detection

1.4.3 Result Visualization

The platform provides multiple forms of real-time visual feedback to enhance interpretability:

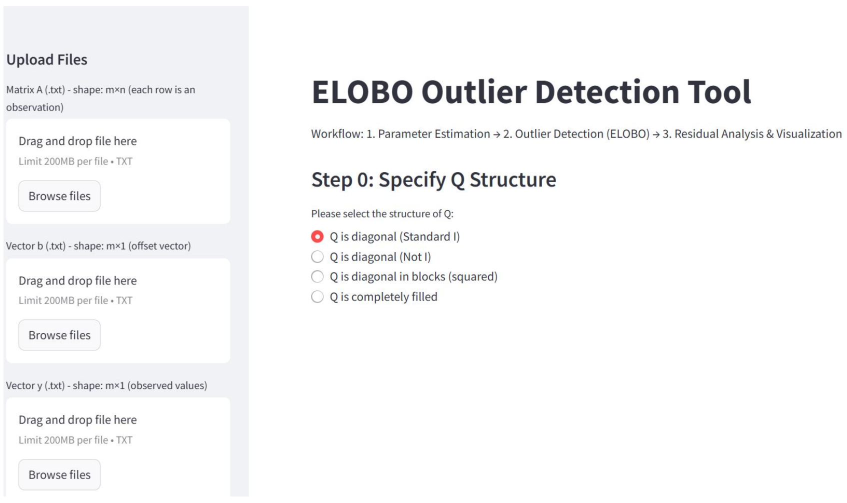
Residual Plot: Compares actual values to predicted values, with residual bars indicating estimation error.

Block Structure Heatmap: Displays the blocks found in the Q matrix, differentiated by color areas in the heatmap.

Outlier Table: Shows the influence metrics for each block (omega norm, delta norm), detection status, and export options.

Downloadable CSV: Users can output the results for further analysis.

These visualizations help users trace the correlations in the data and their impact on estimation accuracy, as well as identify influential subsets of observations.



Upload Files

Matrix A (.txt) - shape: m×n (each row is an observation)

Drag and drop file here
Limit 200MB per file • TXT
Browse files

Vector b (.txt) - shape: m×1 (offset vector)

Drag and drop file here
Limit 200MB per file • TXT
Browse files

Vector y (.txt) - shape: m×1 (observed values)

Drag and drop file here
Limit 200MB per file • TXT
Browse files

ELOBO Outlier Detection Tool

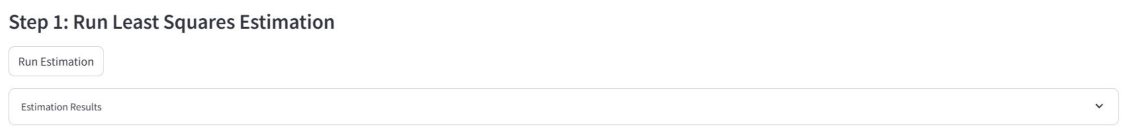
Workflow: 1. Parameter Estimation → 2. Outlier Detection (ELOBO) → 3. Residual Analysis & Visualization

Step 0: Specify Q Structure

Please select the structure of Q:

- [✓ Q is diagonal (Standard I)]
- Q is diagonal (Not I)
- Q is diagonal in blocks (squared)
- Q is completely filled

Figure: Homepage

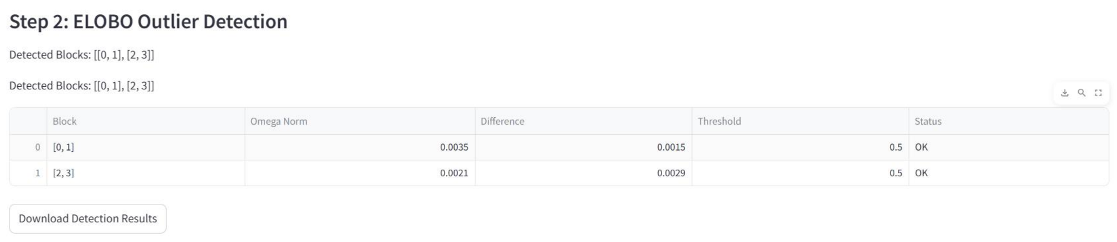


Step 1: Run Least Squares Estimation

Run Estimation

Estimation Results

Figure:LS estimation



Step 2: ELOBO Outlier Detection

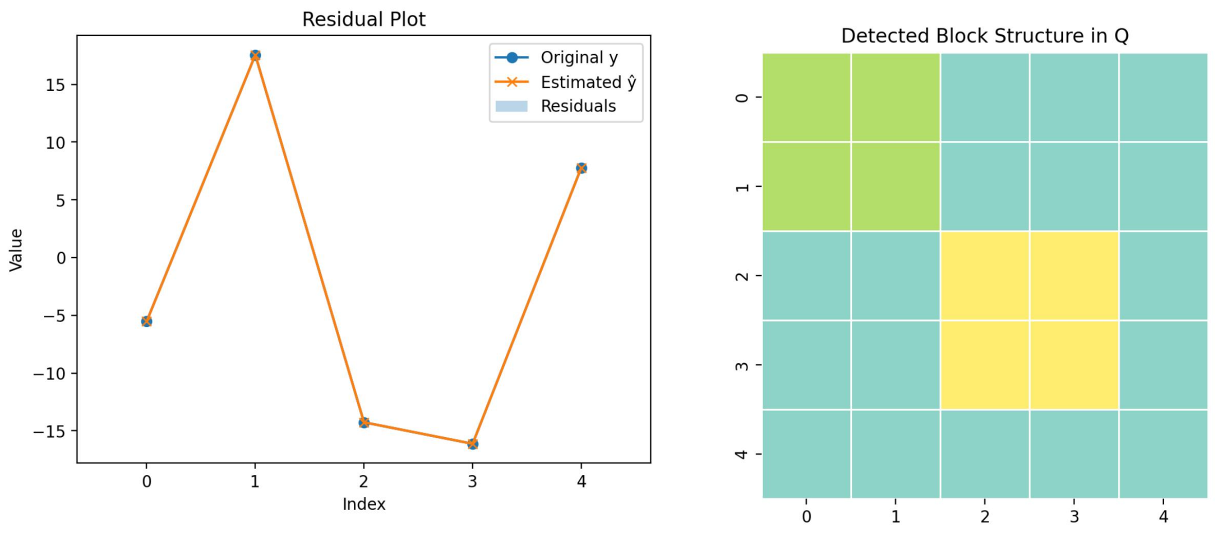
Detected Blocks: [[0, 1], [2, 3]]

Detected Blocks: [[0, 1], [2, 3]]

	Block	Omega Norm	Difference	Threshold	Status
0	[0, 1]	0.0035	0.0015	0.5	OK
1	[2, 3]	0.0021	0.0029	0.5	OK

Download Detection Results

Figure： ELOBO Outlier Detection



Figure： result of LS estimation and Outlier detection

1.5. Implementation Details

1.5.1 Environment Setup

It is recommended to use a virtual environment to manage dependencies and maintain a clean development workspace. The project supports Python 3.8+, and it is tested with Python 3.10. Below are the steps to set up the environment:

To run the application, ensure Python 3.8 or higher is installed. Then install the required packages by executing:

```
pip install streamlit numpy pandas matplotlib
```

Once all dependencies are satisfied, launch the app via:

1. Create a virtual environment named venv310

```
python3.10 -m venv venv310
```

2. Activate the virtual environment

On Windows:()

```
venv310\Scripts\activate
```

3. Install the required Python libraries

```
pip install streamlit numpy pandas matplotlib seaborn
```

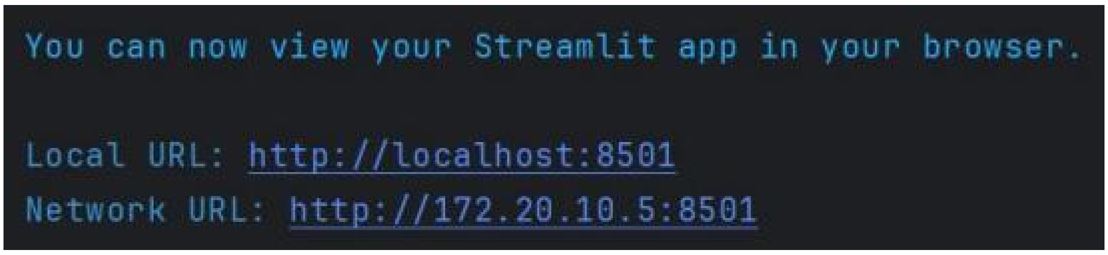
4. Launch the application

```
streamlit run elobo_project.py
```

Once started, the app will open in your default browser, where you can upload matrices and perform the full ELOBO outlier detection workflow.

```
Local URL: http://localhost:8501
```

```
Network URL: http://10.169.145.23:8501
```



```
You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://172.20.10.5:8501
```

Figure: Local& Network URL illustration

1.5.2 Software Tools and Libraries

The project is developed entirely in Python and leverages several open-source libraries for numerical computation, visualization, and user interface rendering. The key tools used are:

·Python 3.8+ – The core language used to develop all functions and scripts.

· Streamlit – Enables the creation of an intuitive, interactive web-based interface.

· NumPy – Provides efficient array and matrix operations, used for all mathematical computation.

·Pandas – Used for handling tabular output and displaying results.

· Matplotlib & Seaborn – Used for plotting residuals, heatmaps, and diagnostics.

1.5.3 File Structure and Functionality

The project is contained in a single Python script file with modularized functions. Below is a breakdown of the core components:

File	Description
elobo_project.py	Main application file containing all logic for data upload, estimation, outlier detection, and visualization.
A.txt / b.txt / y.txt	Input files representing matrix A, vector b, and observation vector y, respectively.
Q.txt (optional)	User-supplied matrix defining the covariance structure between observations.
elobo_results.csv	Optional output file generated after outlier detection for external use.

Functions are grouped logically into:

Data Loading: Reading and verifying matrix shapes.

Matrix Computation: Least squares estimation, Q structure detection.

Outlier Detection:By using the ELOBO algorithm.

Visualization: Implement heatmaps, residual plots, and block diagnostics.

Interface Logic: Streamlit interface layout and state management.

1.5.4 Runtime Instructions (based on README.MD)

```
streamlit run elobo_project.py
```

This will start a local web server and open the application in your default browser. Users can then upload their data matrices (A, b, y, Q), choose the appropriate Q structure, adjust parameters such as sigma² and outlier threshold, and visualize the results interactively.

2.Conclusion and future developments

This section summarizes the conclusions of this project and provides directions for future improvements.

2.1 Conclusion

2.1.1 Summary of Contributions

Through this paper, we introduced an accurate and effective ELOBO implementation for anomaly detection on correlated observations. The main contributions of this paper are as follows:

· Provides an efficient application that can import data structures A, b, y and Q into matrix form, allowing users to choose to use different covariance structures to calculate the least squares method.

· Supports four different Q matrix structures: identity matrix, diagonal matrix, block diagonal matrix, and fully filled matrix, which can handle independent and correlated experimental situations according to user's actual needs.

· Graph-based discovery (DFS) pruning is the basic principle used to continuously extract blocks, so that the system can adaptively learn the correlations in Q.

· Block-based anomaly detection algorithm is a solution that can reduce the full recalculation at each iteration, greatly improving the running time and is faster than simple methods.

· The web interface based on Streamlit is interactive and easy to use, and supports visualizing residuals and related blocks, and supports exporting and downloading results.

2.1.2 Limitations

Although the system is effective, there are still some limitations:

· Currently, it does not support real-time updating of the Q structure after loading, that is, the Q type must be customized by the user before calculation.

· This method is mainly for batch processing and not support online data input.

2.2 Future Improvements

During the development and implementation of the features, we achieved our goals while also identifying areas for improvement. To make the system stronger and easier to use, we plan to enhance the following aspects in the future:

·Enhance algorithm quality to improve the handling of large-scale datasets.

· Achieve modular integration: Convert the core logic into Python modules for use in other applications.

· Statistical tests and multivariate analysis: Perform multiple tests to assess the significance of outliers.

· Enhance user experience: Add matrix editor auto-completion, drag-and-drop file uploads, and step-by-step guides to make the interface more user-friendly.