Data-Driven Investment Project

Group 2

Authors:

Emiliano Della Vecchia (Università L. Bocconi)
emiliano.dellavecchia@master.unibocconi.it

Federico La Penna (Università L. Bocconi)
federico.lapenna@master.unibocconi.it

Federico Rossi(Università L. Bocconi)
federico.rossi6@master.unibocconi.it

Tommaso Zazzaron (Università L. Bocconi)
tommaso.zazzaron@master.unibocconi.it

May 2025

Contents

Introduction

This report addresses Assignment 3: the automatic denoising of variance–covariance matrices. The primary goal is to implement a routine that fits the Marčenko–Pastur probability density function to the empirical correlation matrix of asset returns and then automatically identifies and replaces noisy eigenvalues. Once the noise filtration is complete, a standard portfolio-allocation analysis is performed both before and after denoising.

A sensitivity study is also carried out to investigate how the number of noisy eigenvalues evolves as the portfolio dimension N increases. By examining subsets of assets of increasing size, one can observe the impact of high dimensionality on the sample correlation matrix and demonstrate the essential role of denoising in robust portfolio construction.

1 Theoretical Background

As outlined in the introduction, this report implements an automated denoising procedure for the variance–covariance matrix. This step is critical in portfolio construction, as it produces more stable inverse–covariance estimates and thus yields more reliable portfolio weights out-of-sample.

1.1 Portfolio Optimization Theory

Portfolio optimization techniques relies on performing a portfolio-variance minimization (risk minimization) "maximizing" the expected return of the portfolio. Among the optimization techniques, the most relevant ones are the minimum variance portfolio (MVP) and the maximum sharpe ratio portfolio (MSR), also known as "market portfolio".

MVP solves:

$$w^*_{\text{MV}} = \arg\min_{w \in \mathbb{R}^N} \ \frac{1}{2}\,\text{Var}\big[r_\pi\big]$$
$$\text{s.t.} \quad w^\top \mathbf{1} = 1$$

MSR solves:

$$w^*_{\text{MSR}} = \arg\max_{w \in \mathbb{R}^N} \frac{\mathbb{E}\big[r_\pi\big]}{\sqrt{\text{Var}\big[r_\pi\big]}}$$
$$\text{s.t.} \quad w^\top \mu = 1$$

These optimization problems can be briefly written as follows.

$$w^* = \arg\min_{w \in \mathbb{R}^N} \ \frac{1}{2}\,\text{Var}\big[r_\pi\big]$$
$$\text{s.t.} \quad w^\top a = 1$$

where if $a = \mathbf{1}$ the problem can be reconducted to the MVP one, while if $a = \mu$ to MSR. The optimization problem is faced by solving the associated lagrangian function

$$\mathcal{L}(w, \lambda) = \frac{1}{2}\, w^\top \Sigma\, w \ - \ \lambda\big(w^\top a - 1\big)\ , \lambda \in \ \mathbb{R}$$

By the first-order conditions

$$\frac{\partial \mathcal{L}}{\partial w} = \Sigma\, w \ - \ \lambda\, a, \qquad \frac{\partial \mathcal{L}}{\partial \lambda} = w^\top a \ - \ 1.$$

The solutions are respectively:

$$\text{If } a = \mathbf{1}, \quad w^*_{\text{MV}} = \frac{\Sigma^{-1}\,\mathbf{1}}{\mathbf{1}^\top\,\Sigma^{-1}\,\mathbf{1}},$$
$$\text{If } a = \mu, \quad \tilde{w}^*_{\text{MSR}} = \frac{\Sigma^{-1}\,\mu}{\mu^\top\,\Sigma^{-1}\,\mu}.$$

Where w^*_{MSR} is the not normalized version: If the normalized version is seeked, an adjustment should be done $w^*_{MSR} = \frac{\Sigma^{-1}\mu}{\mathbf{1}^\top \Sigma^{-1}\mu}$. It is important to note that in both cases the optimal weights depend on the inverse covariance matrix. This reliance is critical, since the covariance estimate is notoriously unstable for financial return data and can introduce significant estimation error. This problem will be discussed later in the report.

Numerically speaking, the implementation of this optimization has been performed in python by the following function:

Optimal Portfolio Weights – `optPortNum`

```
def optPortNum(cov, mu=None, flag=1, shortSellingConstraints=False):
    N = cov.shape[0]

    if flag == 1:
        a = np.ones((N, 1))
    elif flag == 2:
        a = mu

    if shortSellingConstraints:
        bnds = tuple((0, 1) for _ in range(N))
    else:
        bnds = None

    EWP = np.ones(N) / N

    output = minimize(objFct,EWP,args=cov, bounds=bnds,
        constraints=({'type': 'eq', 'fun': lambda w: 1 - np.dot(w.T, a)})
        ,options={'maxiter': 10000})

    if output['success']:
        wstar = output['x']
    else:
        wstar = None
    return wstar / wstar.sum()
```

This implementation doesn't exploit any closed weights formula, but performs a minimization by the built-in functionality in scipy.optimize, minimize.
In addition, two functions are built to retrieve the yearly essential Var-Cov matrix and the expected returns of the return time series. Moreover, a function that performs the portfolio statistics (portfolio return μ_p, variance of portfolio and portfolio Sharpe-Ratio) is reported. Here below are shown the two

functions:

Moment Estimation – `getMoments`

```
def getMoments(prices_fct):

    returns_fct = (prices_fct / prices_fct.shift(1))[1:] - 1
    mu_fct  = returns_fct.mean() * 250
    cov_fct = returns_fct.cov()  * 250
    return mu_fct, cov_fct
```

Portfolio Statistics – `ptf_stats`

```
def ptf_stats(weights_fct, mu_fct, cov_fct, rf_fct):

    ptf_mu_fct  = np.dot(weights_fct.T, mu_fct)
    ptf_var_fct = np.dot(weights_fct.T, np.dot(cov_fct, weights_fct))
    ptf_std_fct = np.sqrt(ptf_var_fct)
    ptf_SR_fct  = (ptf_mu_fct - rf_fct) / ptf_std_fct
    return ptf_mu_fct, ptf_var_fct, ptf_SR_fct
```

1.2 Denoising Theory

This section explains the motivation for applying covariance-matrix denoising. In empirical correlation matrices, some eigenvalues often lie very close to zero, which renders the inverse covariance Σ^{-1} highly unstable. Since portfolio weights are computed as a function of Σ^{-1}, and since Σ^{-1} is proportional to the inverse of the determinant (which is $det(\Sigma) = \prod_i \lambda_i$), these near-zero eigenvalues introduce excessive noise into the allocation. Denoising addresses this issue by replacing the smallest eigenvalues with an average value. The resulting "cleaned" Σ yields a much more stable inverse and produces portfolio weights that are both more robust and economically coherent.

The denoise procedure starts by converting the sample variance-covariance matrix into the sample correlation matrix, to which the spectral decomposition will be applied.

Spectral Decomposition Let C be a symmetric matrix. Then, $\exists$ U which is orthogonal ($UU^T = U^TU = 1$) such that:

$$C = U\,\Lambda\,U^\top$$

where U is the matrix of eigenvectors and Λ is the diagonal matrix containing the eigenvalues $\lambda_1, \ldots, \lambda_N$.

Having performed the decomposition, the analysis of the eigenvalues can be performed. In order to

denoise the matrix it is essential to identify the number k of informative eigenvalues $\lambda_i > \lambda_+$, where λ_+ is the upper edge of the Marčenko–Pastur distribution. Specifically, $\lambda^+ = \sigma^2 \left(1 + \sqrt{\frac{N}{T}}\right)^2$. Where σ^2 is computed through the Marcenko-Pastur fitting.

Fitting Empirical Eigenvalues and Theoretical Marcenko-Pastur Distribution:
By fitting the empirical distribution and the Marcenko-Pastur (MP), the threshold λ^+ is setted and the small noisy eigenvalues can be replaced by a constant one $\bar{\lambda}$. Recalling that the MP distribution is

$$f(\lambda) = \begin{cases} \frac{T}{N} \cdot \frac{\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)}}{2\pi\lambda\sigma^2} & \text{if } \lambda \in [\lambda_-, \lambda_+] \\ 0 & \text{else} \end{cases}$$

and that the empirical distribution can be computed numerically as the Euler central difference of the empirical cumulative distribution function:

$$\hat{f}(\lambda) \approx \frac{\hat{F}(\lambda + \epsilon) - \hat{F}(\lambda - \epsilon)}{2\epsilon}$$

the minimization is computed by the MSE loss:

$$\hat{\sigma}^2 = \arg\min_{\sigma^2 \in \mathbb{R}^+} \sum_i (f(\lambda_i)_{MP} - f(\lambda_i)_{emp})^2$$

Here below are shown the algorithms function that are used for the computation of σ^2.

Spectral Decomposition – `getSpectralDec`

```
def getSpectralDec(A):
    eVal, eVec = np.linalg.eigh(A)
    indices = np.flip(np.argsort(eVal))
    eVal = eVal[indices]
    U = eVec[:, indices]
    lamb = np.diag(eVal)
    return lamb, U
```

Empirical PDF via Finite Difference – `empirical_lambda`

```
def empirical_lambda(lamb, pts):
    lamb_int = np.linspace(np.min(lamb), np.max(lamb), pts)
    step     = (np.max(lamb) - np.min(lamb)) / (pts - 1)
    epsilon  = step / 2
    pdf_hat = np.zeros(pts)
    for i in range(pts):
        cdf_up   = np.sum(lamb <= lamb_int[i] + epsilon) / len(lamb)
        cdf_down = np.sum(lamb <= lamb_int[i] - epsilon) / len(lamb)
        pdf_hat[i] = (cdf_up - cdf_down) / (2 * epsilon)
    return lamb_int, pdf_hat
```

Marčenko–Pastur PDF – `MarPasPDF`

```
def MarPasPDF(sigma2, q, x):
    lam_minus = sigma2 * (1 - np.sqrt(1/q))**2
    lam_plus  = sigma2 * (1 + np.sqrt(1/q))**2
    pdf       = np.zeros_like(x)
    mask      = (x >= lam_minus) & (x <= lam_plus)
    pdf[mask] = q * np.sqrt((lam_plus - x[mask]) * (x[mask] - lam_minus)) \
                / (2 * np.pi * sigma2 * x[mask])
    return pdf, lam_minus, lam_plus
```

MSE Loss Function – `loss`

```
def loss(sigma2, lambdas, q, pts, bw_method=None):
    grid, pdf_emp = empirical_lambda(lambdas, pts=pts)
    pdf_mp        = MarPasPDF(sigma2, q, grid)[0]
    return np.sum((pdf_emp - pdf_mp)**2)
```

Having computed the optimal σ^2 and therefore the threshold, λ^+, the new $\tilde{\Lambda}$ vector can be computed. Because the eigenvalues are sorted from the biggest to the lowest, by denoting $\#\lambda_i < \lambda^+$ with k, the following rule holds:

$$\widetilde{\lambda}_i = \begin{cases} \lambda_i & \text{if } i \leq k \\ \\ \dfrac{1}{N-k} \displaystyle\sum_{i=k+1}^{N} \lambda_i & \text{if } i > k \end{cases}$$

This replacement preserves the structure of the informative part of the spectrum, while averaging the noisy eigenvalues to reduce estimation noise.

An intermediate matrix $\widetilde{C}$ is then reconstructed:

$$\widetilde{C} = U\,\widetilde{\Lambda}\,U^{\top}$$

Since $\widetilde{C}$ may not have unit entries on the diagonal, it is not yet a proper correlation matrix. To ensure unit diagonal entries, a renormalization is applied:

$$C_1 = D^{-1/2}\,\widetilde{C}\,D^{-1/2}$$

where $D = \mathrm{diag}(\widetilde{C})$ is the diagonal matrix whose diagonal entries are those of $\widetilde{C}$. The resulting matrix C_1 is symmetric and has unit diagonal entries.
Finally, the denoised matrix reconstruction's algorithm is reported here below.

Covariance Denoising – `denoisedMatrix`

```
def denoisedMatrix(Delta_matrix, U_matrix, n_goodEig):
    N  = Delta_matrix.shape[0]
    vals = np.diag(Delta_matrix).copy()
    vals[n_goodEig:] = vals[n_goodEig:].mean()
    D_denoised = np.diag(vals)
    M_denoised = U_matrix @ D_denoised @ U_matrix.T
    norm_mat   = np.linalg.inv(np.diag(np.sqrt(np.diag(M_denoised))))
    return norm_mat @ M_denoised @ norm_mat.T
```

2 In-Sample and Out-Of-Sample splitting, Covariance-Correlation Bridge and Distribution Fitting

2.1 Empirical vs Theoretical MP PDF Results

Once the value of σ^2 that best fits the empirical eigenvalue distribution to the theoretical Marčenko–Pastur (MP) distribution was identified, the analysis proceeds to evaluate the quality of the fit and to determine which eigenvalues carry information versus noise.
After performing the minimization of the MSE loss between the Marcenko-Pastur pdf and the empirical distribution the optima σ^2 rised.

$$\sigma^2 = 0.44648438$$

Then, the first comparison involved plotting the theoretical MP probability density function (PDF) alongside the empirical eigenvalues distribution. The fit was achieved by optimizing the MP parameters λ_- and λ_+, which in this case were found to be approximately:

- $\lambda_- \approx 0.1962$
- $\lambda_+ \approx 0.7982$

The empirical distribution aligned closely with the theoretical MP curve, indicating that the optimization process was effective and that the bulk of the eigenvalues is consistent.

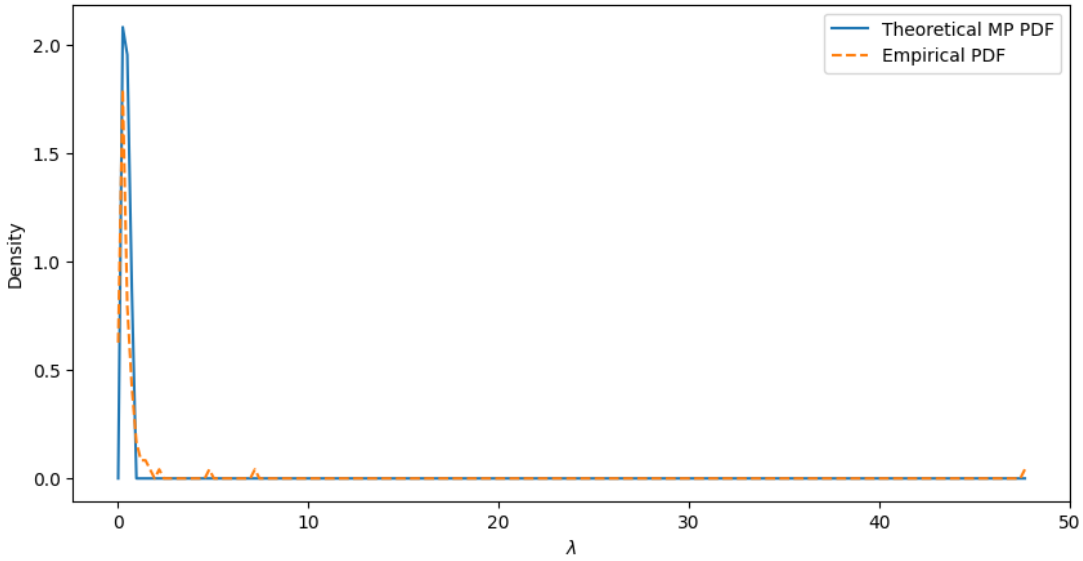

Figure 1: Theoretical MP PDF vs Empirical PDF

2.2 Histogram Comparison of Empirical Eigenvalues

To further validate the fit, a histogram of the empirical eigenvalues was plotted and overlaid with the theoretical MP PDF. The histogram revealed that a substantial portion of the eigenvalue mass falls within the $[\lambda_-, \lambda_+]$ interval, as prescribed by the Marčenko-Pastur law.

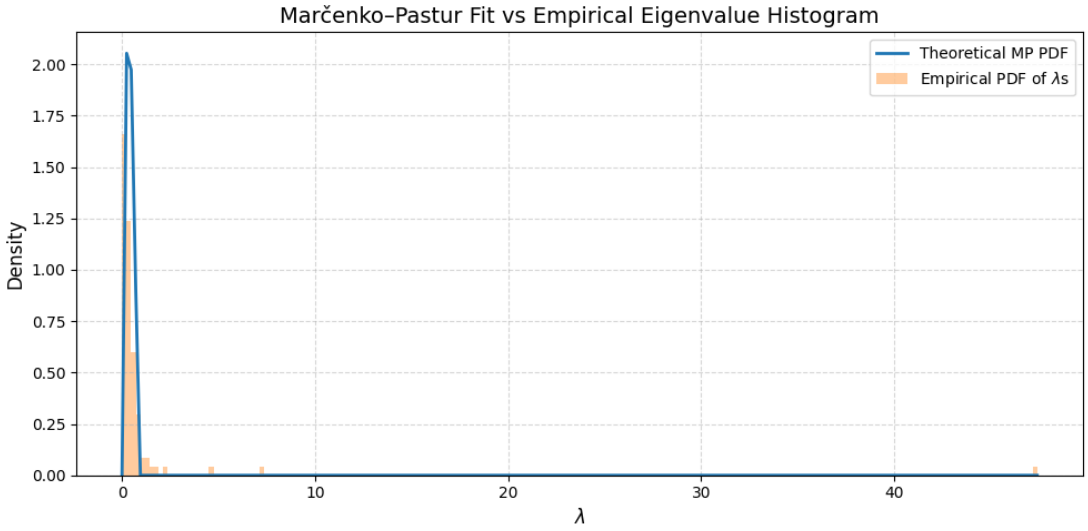

Figure 2: Histogram of empirical eigenvalues and fitted MP distribution

2.3 Empirical Eigenvalue Distribution

Lastly, the standalone empirical eigenvalue distribution was visualized to emphasize the heavy concentration of eigenvalues near zero, with a few outliers well beyond the theoretical upper bound λ_+. These outliers may potentially correspond to genuine information and will be the focus of subsequent denoising strategies.

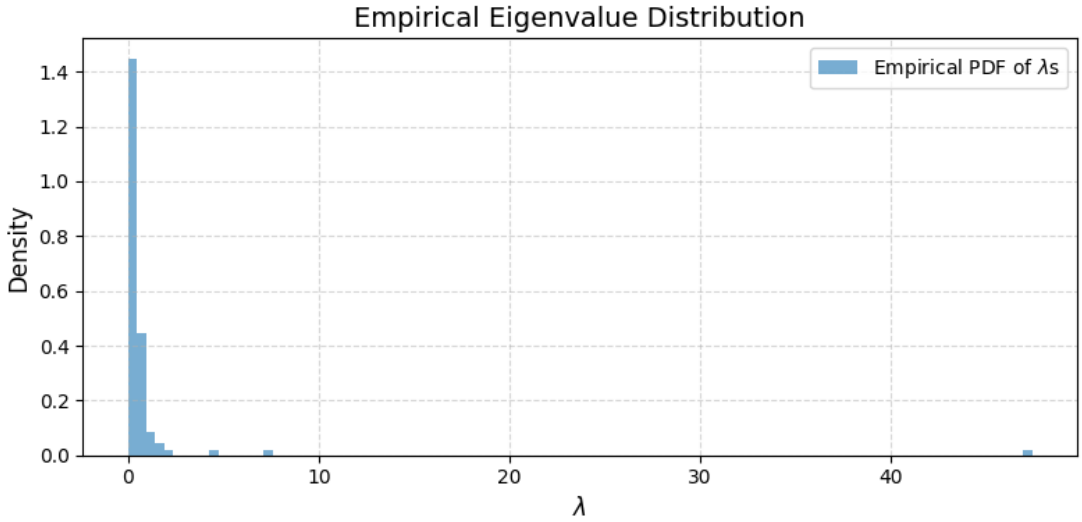

Figure 3: Empirical eigenvalue distribution

2.4 Classification of Eigenvalues: Informative vs Noisy

In this subsection, the eigenvalues of the empirical covariance matrix were classified into informative and noisy components.

This classification is based on the theoretical upper bound λ_+ of the Marčenko–Pastur spectrum, given by $\lambda_+ = \sigma^2 \left(1 + \sqrt{\frac{N}{T}}\right)^2$, where N is the number of assets and T is the number of observations.

Eigenvalues greater than λ_+ are considered to contain informative signal, while those below this threshold are likely due to noise.

The number of informative and noisy eigenvalues was computed as follows:

- Number of informative eigenvalues: $n_{\text{good}} = 15$
- Number of noisy eigenvalues: $n_{\text{bad}} = 85$

This empirical result suggests that only a small fraction (15%) of the eigenvalues are significantly larger than the noise threshold and may carry useful information for portfolio construction or risk estimation. The remaining 85% are presumed to be dominated by estimation noise and will be candidates for denoising in the next steps.

3 Variance-Covariance Matrix Denoising

The goal of this section is to construct a denoised correlation matrix C_1 that is numerically stable. In particular, the denoising process aims to "eliminate" extremely small eigenvalues from the correlation matrix C, which can lead to numerical instability in matrix inversion and degrade portfolio optimization performance.

After applying the denoising procedure, the resulting correlation matrix is obtained.

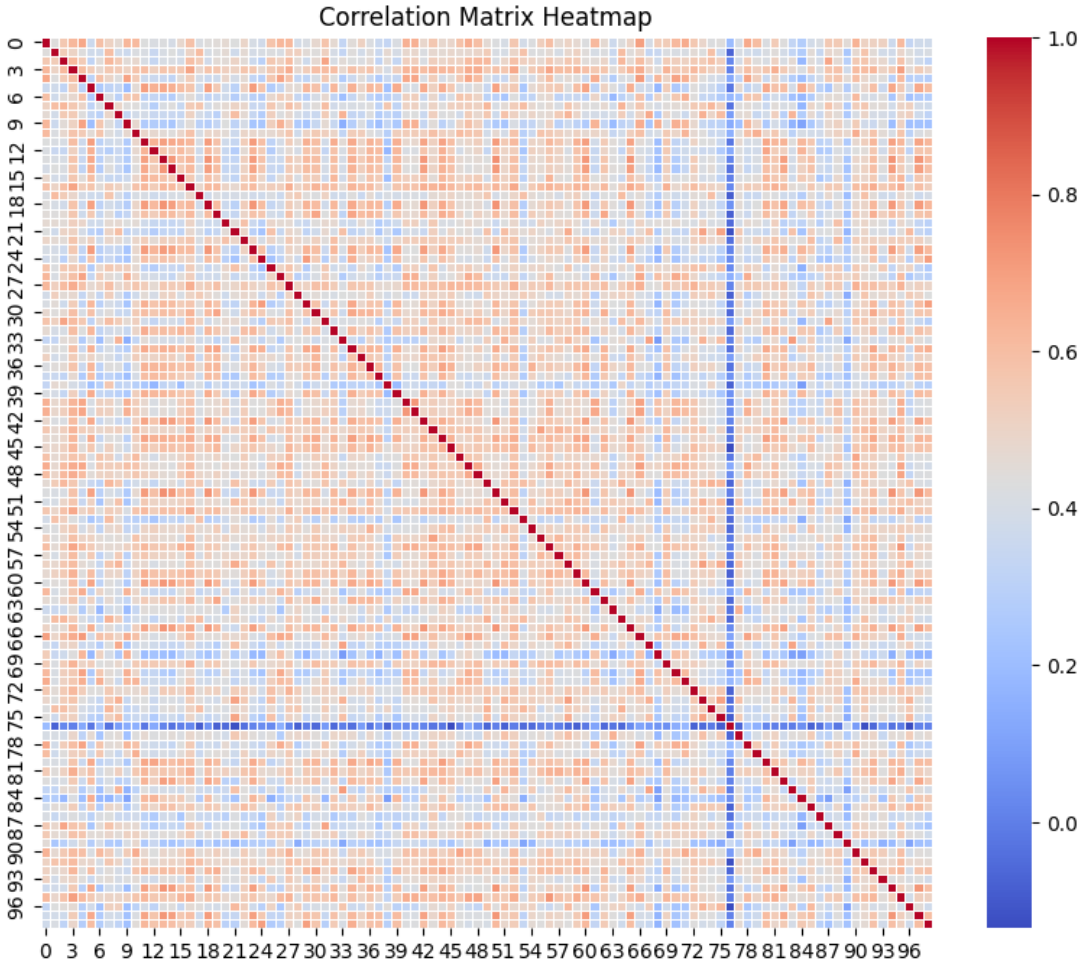

Figure 4: Heatmap of the denoised correlation matrix C_1

3.1 Reconstruction of the Denoised Covariance Matrix

Once the denoised correlation matrix C_1 is obtained, it can be transformed back into a variance-covariance matrix using the asset volatilities. This is done via the standard transformation:

$$\Sigma_{\text{denoised}} = \text{diag}(\sigma) \cdot C_1 \cdot \text{diag}(\sigma)$$

where $\sigma = \sqrt{\text{diag}(\Sigma)}$ is the vector of standard deviations estimated from the empirical covariance matrix.
This denoised covariance matrix Σ_{denoised} can now be used for robust portfolio optimization, risk management, or further analysis.
Here below is shown the algorithm implementation.

Correlation to Covariance – `corr2cov`

```
def corr2cov(corr, stds):
    return corr * np.outer(stds, stds)
```

4 Portfolio Allocation Analysis

In this section, key portfolio statistics are computed and compared across several portfolio construction methods. The performance of each portfolio is evaluated **out-of-sample** evaluating the expected Return, variance and Sharpe Ratio.

The analysis includes the following five portfolios:

1. Equally Weighted (Naive) Portfolio
2. Global Minimum Variance Portfolio (before denoising)
3. Maximum Sharpe Ratio Portfolio (before denoising)
4. Global Minimum Variance Portfolio (after denoising)
5. Maximum Sharpe Ratio Portfolio (after denoising)

This comparison aims to assess the impact of the denoising procedure on portfolio performance.

Here below the numerical result from the analysis is performed.

Portfolio	Expected Return	Variance	Sharpe Ratio
Naive	0.175598	0.014728	1.443489
Min Var	-0.370551	0.041071	-1.830499
Max Sharpe	-0.425884	0.172483	-1.026470
Min Var - denoised	-0.034139	0.011418	-0.323420
Max Sharpe - denoised	0.394299	0.202859	0.874512

Table 1: Out-of-sample performance statistics

To better visualize the comparison, the metrics were plotted in a grouped bar chart.

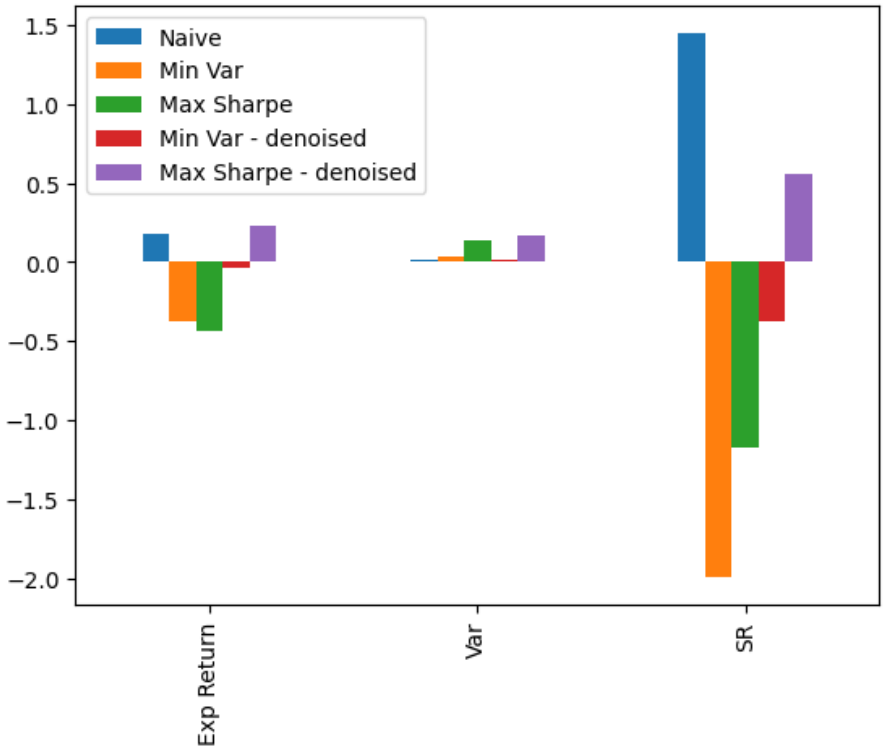

Figure 5: Bar chart of portfolio statistics

From the results, it is evident that the Max Sharpe-denoised portfolio performs significantly better than its noisy counterpart in both expected return and Sharpe ratio, by maintaining an almost equal variance. Conversely, the non-denoised portfolios, especially the Minimum Variance and Maximum Sharpe, yield negative out-of-sample Sharpe ratios, reflecting instability likely caused by noise in the empirical covariance estimation.

The denoising procedure clearly contributes to a more robust portfolio construction by mitigating the effects of sampling noise in the covariance matrix.

5 Sensitivity Analysis

To assess the robustness and relevance of the denoising procedure in varying dimensional contexts, a sensitivity analysis is performed. The goal is to evaluate how the number of noisy eigenvalues evolves as the number of assets N increases.

5.1 Methodology

The analysis iteratively constructs the correlation matrix from an increasing number of assets. Starting from a small subset, one additional stock is included at each iteration, and the Marčenko–Pastur procedure is reapplied to classify the eigenvalues. Specifically, at each step:

1. A correlation matrix is computed using data from N assets.
2. Spectral decomposition yields the set of eigenvalues.
3. The upper bound λ_+ of the MP distribution is determined via numerical optimization.
4. The number of eigenvalues below λ_+ is counted and classified as noisy.

This process is repeated until all assets in the dataset are included.
Here below is shown the algorithm that performs such routine.

Denoising Across Expanding Asset Sets

```
eig_noisy = []
for i in range(len(tickers)):
    price = prices.copy()
    price = price.iloc[:, :i+1]
    returns = (price / price.shift(1))[1:] - 1
    price_training = price['2018-1-1':'2021-6-30']
    mu_training, cov_training = getMoments(price_training)
    corr_training = cov2corr(cov_training)
    Lambda_training, U_training = getSpectralDec(corr_training)
    eigvals = np.diag(Lambda_training)
    T, N = price_training.shape
    q = T / N, pts = 200, ig = 0.3
    bnds = [(0, None)]
    sigma2 = minimize(loss,ig, args=(eigvals, q, pts),
        method='Nelder-Mead',bounds=bnds).x
    emp_lamb, emp_pdf = empirical_lambda(eigvals, pts=200)
    pdf_mp, a, b = MarPasPDF(sigma2, q, emp_lamb)
    n_badeig = np.sum(np.diag(Lambda_training) <= b)
    eig_noisy.append(n_badeig)
```

5.2 Results

The resulting curve shows how the number of noisy eigenvalues increases with N, as expected. The rise is roughly monotonic, with minor fluctuations due to the randomness of empirical eigenvalues.

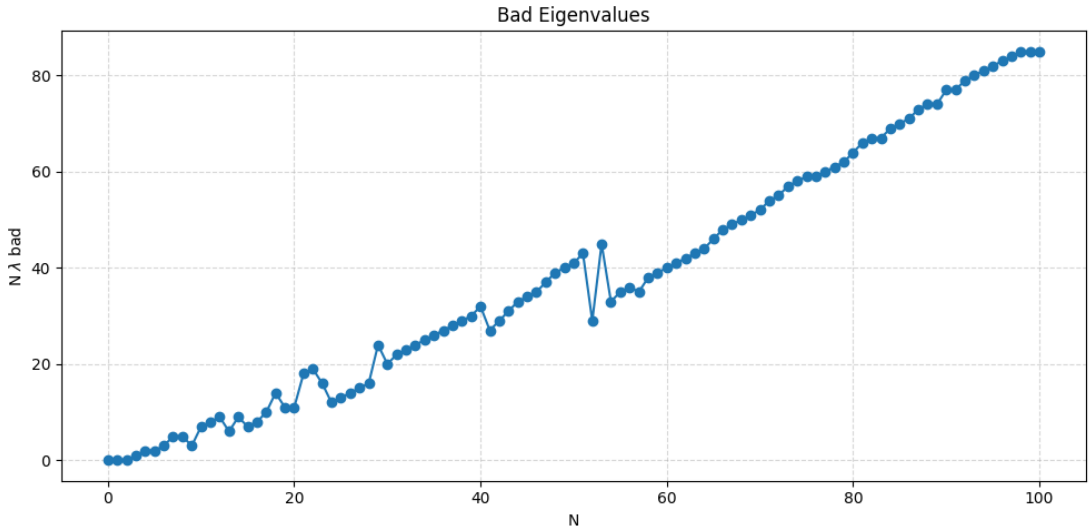

Figure 6: Evolution of the number of noisy eigenvalues

The observed increasing trend highlights the growing impact of noise in high-dimensional correlation matrices. As N becomes large relative to the number of time observations T, more eigenvalues fall within the Marčenko–Pastur bulk, which reinforces the necessity of denoising procedures in large-scale portfolio construction problems.

This analysis confirms that in practical financial applications with large universes of assets, denoising is not just beneficial but essential to reduce the estimation error inherent in empirical covariance matrices.