Erasmus University Rotterdam

Erasmus School of Economics

Sparse index clones via the sorted ℓ_1 norm

Bachelor Thesis: FEB63007

Supervisor: Dr. N.W. Koning

Second Assessor: Dr. C. Zhou

Author: Albert Liu

Student number: 500507

July 3, 2022

Abstract

Index tracking (IT) is a way to create a passive replication portfolio to track certain benchmarks, such as a market index or hedge fund. Kremer et al. (2022) propose a Sorted ℓ_1 Penalized Estimator (SLOPE) estimator in the IT framework that creates sparse and stable mimicking portfolios to improve cost efficiency. The SLOPE estimator requires a non-decreasing lambda sequence, which is chosen arbitrary. In this paper, we investigate if the SLOPE estimator produces robust results by alternating the used parameters and mainly the form of the lambda function. Moreover, we elaborate on possible problems that could exist in the no-short sale area when the lambda components of SLOPE become approximately equal to each other. We obtain daily return observations of the S&P100 and S&P500 from 31 December 2004 to 29 January 2016, and use a simulation study and a rolling window approach proposed by Kremer et al. (2022) to conduct our research. We find that the SLOPE estimator is dependent on the magnitude of the lambda parameters as well as the steepness of the lambda function. Lastly, when the SLOPE penalty converges to the LASSO penalty, it also suffers in the no-short sale area just as LASSO and loses its grouping properties, given an imposed budget constraint.

Table of contents

1 Introduction

Index tracking (IT) is a way to create a passive replication model to track certain benchmarks, such as a market index or hedge fund. However, mimicking an index can be costly as you take positions in every constituent of the index while also re-balancing the composition of your tracking portfolio frequently. In addition, Kremer et al. (2022) mention that in some cases, it is not even possible to completely track an index since small illiquid assets might not be allowed to be traded in large volumes.

To overcome these problems, Kremer et al. (2022) introduce the Sorted ℓ_1 Penalized Estimator (SLOPE), which penalizes the asset weights with a sorted ℓ_1 norm that is sorted according to its importance to the benchmark index. Therefore, SLOPE uses a sequence of tuning parameters and solves the optimization problem by minimizing the squared ℓ_2 norm of the difference between the benchmark index and the tracking portfolio. This leads to tracking portfolios that are sparse (i.e. have fewer active positions) and stable (i.e. have a lower turnover) while still mimicking the specific index. SLOPE is based on the concept of grouping assets with similar partial correlations to an index and assigning the same coefficient values to these assets.

As was mentioned in Kremer et al. (2020), SLOPE yields optimal portfolios with good out-of-sample risk and return performance properties, by reducing the overall turnover, through more stable asset weight estimates. In addition, SLOPE is a sparse solution and therefore less costly than the full replication strategy. These features are advantageous for investors, portfolio managers, and anyone else seeking exposure to a particular broad market index. Moreover, benchmark indices are easier to invest in than actively managed funds, and tracking indices can be added to your portfolio for diversification purposes.

Our research extends on the work of Kremer et al. (2022) in the following two directions. Firstly, we replicate the simulation- and empirical study that was done by Kremer et al. (2022). As not all hyper parameters are provided in the paper, the replicated study yields different results. Nonetheless, the results were of the same order of magnitude. Since the results may depend on the hyper parameters selected, our first research question is as follows:

RQ1 : *Are the weight positions in the tracking portfolios obtained from the SLOPE estimator of Kremer et al. (2022) sensitive to different lambda sequences?*

Through simulation, we show that the weight composition is dependent on the magnitude of the lambda components and the steepness of the lambda sequence.

Secondly, a well known penalty is the Least Absolute Shrinkage and Selection Operator (LASSO) introduced by Tibshirani (1996). LASSO penalizes the weight vector using an ℓ_1 norm. Given an imposed budget constraint (i.e. $\sum_{i=1}^{K} w_i = 1$, where w_i represents the ith asset weight) and the long-only constraint (i.e. $w_i \geq 0$), the LASSO penalty is always equal to one. Consequently, the LASSO penalty is of no use to the optimization problem. Due to the fact that LASSO is a special case of SLOPE when all lambda parameters are equal, this leads to the second research question.

RQ2 : *Does the SLOPE estimator also suffer in the no short-sale area given an imposed budget* *constraint as it converges to LASSO?*

Using a simulation analysis, we find that when the SLOPE penalty converges to the LASSO penalty, it loses its grouping properties and therefore also becomes stuck in the no-short sale area just as the LASSO penalty.

In our paper, we use daily return observations of the S&P100 and S&P500 from 31 December 2004 to 29 January 2016 ($T = 2890$ observations) gathered by Datastream. We apply a rolling window approach to analyze the out-of-sample performance of our tracking portfolio for different lambda sequences and different orders of magnitude for the lambda components. Finally, we use a simulation study to determine whether the SLOPE estimator encounters the same difficulties as the LASSO estimator when the lambda components become nearly equal.

2 Literature

This section elaborates more on different penalties in the IT framework, focusing primarily on the SLOPE penalty. Next, we discuss the theoretical properties of SLOPE. In this section, we use the same notation that was done by Kremer et al. (2022) and it differs only if stated otherwise.

In the IT framework, an investor wants to create a sparse and stable portfolio that best tracks the given benchmark. In order to do so, consider the $T \times 1$ benchmark return vector $\mathbf{Y} = [Y_1, Y_2, \ldots, Y_T]'$, the $T \times K$ return matrix $\mathbf{R} = [\mathbf{r^1}, \ldots, \mathbf{r^k}]$, where $\mathbf{r^i}$ is the $T \times 1$ return vector for asset i and the $K \times 1$ weight vector $\mathbf{w} = [w_1, \ldots, w_k]'$. To mimic the given benchmark as closely as possible, we introduce an OLS regression framework and minimize the following penalized tracking error measure subject to a budget constraint:

$$\begin{aligned} &\arg \min_{\mathbf{w} \in \mathbb{R}^k} \frac{1}{2} \ ||\mathbf{Y} - \mathbf{R}\mathbf{w}||_2^2 + \rho_\lambda(\mathbf{w}) \\ &\text{s.t.} \ \ \mathbf{1}^\mathsf{T}\mathbf{w} = \mathbf{1}, \end{aligned} \tag{1}$$

where $\rho_\lambda(\mathbf{w})$ is a certain penalty function and $\mathbf{1}$ is a $K \times 1$ vector of ones. According to DeMiguel et al. (2009), regularization methods taking the form of a constraint in the norm of the weight vector, have found widespread attention in portfolio selection in the last 10 years, to create sparse and stable allocations. An often used penalty that has these theoretical properties is the LASSO penalty, which restricts the weight with an ℓ_1 norm. However, Bondell and Reich (2008) state that LASSO selects randomly from equally correlated assets and more importantly, DeMiguel et al. (2009) mention that LASSO is of no use in the no-short sale area, given an imposed budget constraint. Several other penalties have been created to outwit these disadvantages of LASSO. As stated by Kremer et al. (2022), non-convex regularization methods are able to produce clones with fewer active positions than LASSO, resulting in lower transaction costs for investors. However, Giuzio (2017) mention that the optimization problem becomes NP-hard and presents computational challenges, especially in high-dimensional settings. Consequently, we seek convex and therefore not NP-hard regularization methods.

We introduce the SLOPE estimator that is proposed by Bogdan et al. (2015). As mentioned briefly in

Section 1, SLOPE is a convex program that penalizes estimated weights based on their rank magnitude. It has a grouping property and assigns equal weights to assets within a given group. Statistically, it finds the solution to the optimization problem in (1), where the penalty function $\rho_\lambda(\mathbf{w})$ is equal to:

$$\begin{aligned} &\rho_\lambda(\mathbf{w}) = \sum_{j=1}^{K} \lambda_j |w|_{(i)} = \lambda_1 |w|_{(1)} + \lambda_2 |w|_{(2)} + ... + \lambda_k |w|_{(k)} \\ &\text{s.t.} \quad \lambda_1 \geq \lambda_2 \geq ... \geq \lambda_K \geq 0 \\ &|w|_{(1)} \geq |w|_{(2)} \geq ... |w|_{(k)}. \end{aligned} \tag{2}$$

Here, $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_K]$ is a non-decreasing lambda sequence and $|w_i|$ is the ith largest weight in the estimated weight vector $\mathbf{w}$. The difference between SLOPE and other (non)-convex penalties is that decreasing lambda parameters are chosen rather than a single lambda parameter. In addition, LASSO is a special case of SLOPE when all lambda parameters are set to the same value.

Kremer et al. (2022) provide new theoretical results of SLOPE, which show that the grouping provided by SLOPE is not solely due to the strong correlation between the assets, but it is driven by the similarity of partial correlations of different assets with the respective benchmark. We include two theorems to show this. First, Theorem 1 states that the number of non-zero coefficients is dependent on the return matrix $\mathbf{R}$.

Theorem 1 (Theorem 2.1, Kremer et al. (2022)) *Let $\mathbf{R}$ be the $T \times K$ matrix of benchmark constituents, $\mathbf{Y}$ be the T-dimensional vector of benchmark returns and assume that $\hat{\mathbf{w}}$ is an unique solution to the following optimization problem,*

$$\begin{aligned} &\arg\min_{\mathbf{w} \in \mathbb{R}^k} \frac{1}{2} \; ||\mathbf{Y} - \mathbf{R}\mathbf{w}||_2^2 + \rho_\lambda(\mathbf{w}) \\ &\text{s.t.} \;\; \mathbf{1}^\mathsf{T}\mathbf{w} = \mathbf{1}, \end{aligned} \tag{3}$$

where $\lambda_1 \geq \ldots \geq \lambda_k \geq 0$. If the return matrix $\mathbf{R}$ is of rank $k \leq K$, then the vector $|\hat{\mathbf{w}}|$ contains at most k different non-zero values.

This means that the weight of assets that are linear combinations of other assets is set to zero by the SLOPE penalty. Additionally, it is possible that more asset weights are put to zero if the penalty becomes stronger. Furthermore, Theorem 2 displays the grouping property of SLOPE with respect to the lambda sequence and the correlations between the assets and the residuals:

Theorem 2 (Theorem 2.2, Kremer et al. (2022)) *Let $\hat{\mathbf{w}}$ be the optimal estimated weight vector using SLOPE. Moreover, suppose that all columns of $\mathbf{R}$ have been standardized to the same norm, namely it holds that $d := ||\mathbf{R_1}|| = \ldots = ||\mathbf{R_K}||$ and that the solution satisfies $\hat{\mathbf{w_1}} \geq \ldots \geq \hat{\mathbf{w_K}}$. Then, for any $i \in 1, \ldots, K-1$, it holds that*

$$\hat{w_i} > \hat{w}_{i+1} \Longrightarrow \mathbf{R_i^T} r_p - \mathbf{R_{i+1}^T} r_p \geq \lambda_i - \lambda_{i+1}, \tag{4}$$

where $r_P := \mathbf{Y} - R_{\setminus i,i+1} \hat{w}_{\setminus i,i+1}$ and $R_{\setminus i,i+1}$ and $\hat{w}_{\setminus i,i+1}$ are obtained by removing the ith and $i+1$st columns of $\mathbf{R}$ and elements of $\hat{\mathbf{w}}$.

According to Kremer et al. (2022), $\mathbf{R_i^T} r_p$ measures the correlation between the ith asset and the residual given by r_P, where for the prediction of the benchmark return, the assets i and $i+1$ are excluded. Therefore, it represents the partial correlation of asset i with the benchmark return $\mathbf{Y}$. Consequently, Theorem 2 states that assets with a similar partial correlation to the benchmark $\mathbf{Y}$ will be more likely grouped together and assigned the same weight. Besides, it also shows that the gap between consecutive lambda parameters influences the grouping structure of SLOPE.

Accordingly, the SLOPE estimator could be dependent not only on the magnitude of the lambda sequence but also the form of the sequence, both of which increase the median gap between consecutive lambda parameters. Therefore, it seems that SLOPE is sensitive to the lambda sequence that is selected. This could lead into problems such as assets that are grouped together despite having a low correlation, or results that are not robust in general. Kremer et al. (2022) choose their lambda sequence based on Bogdan et al. (2015), which controls for the False Discovery Rate (FDR). As the shape and magnitude of the lambda sequence may be relevant to the SLOPE solution, and as the current lambda sequence is fairly arbitrary, we use in total eight different lambda functions to check the sensitivity of the SLOPE results from Kremer et al. (2022).

3 Replication

Kremer et al. (2022) perform a simulation analysis by using a data generating process that has a predefined grouping structure to illustrate the theoretical properties of SLOPE. This section first introduces their simulation study and reproduces their simulation and out-of-sample results. Then, to get an idea of which lambda sequences to use for our extension, we consider a two-dimensional asset universe. We use the code from Kremer et al. (2022) to start with and complete the code by ourselves to run the numerical analysis using MATLAB version R2019b. See Appendix B for a guide to our code.

3.1 Simulation analysis

The simulation study done by Kremer et al. (2022) solves the optimization problem in (1) by simulating data for the return matrix $\mathbf{R}$. The return matrix $\mathbf{R}$ is obtained by considering an underlying risk factor structure, where assets belonging to the same group are exposed to the same subset of factors. More precisely, the asset returns are a linear combination of these risk factors:

$$\mathbf{R} = \mathbf{F} \times \mathbf{B} + \epsilon, \tag{5}$$

where $\mathbf{R}_{T\times K}$ is the return matrix, $\mathbf{F}_{T\times S} = [f_1 f_2 ... f_S]$ is the $T\times S$ factor matrix, $\mathbf{B}_{S\times K}$ is the $S\times K$ loading matrix and ϵ is a $T \times K$ matrix of normally distributed error terms. T is the number of observations, S is the total number of risk factors and K is the total number of assets. In our simulation study, the following values are assigned to these variables: $T = 500, S = 3$ and $K = 99$. Furthermore, the underlying risk factors are drawn from an independent multivariate standard normal $\mathcal{N}(0, I_{S\times S})$ distribution, with $I_{S\times S}$ being an identity matrix. The error terms are drawn from a multivariate normal distribution $\mathcal{N}(0, 0.05 \times I_{K\times K})$ and are independent from each other as well as from each of the risk factors. Lastly, the loading matrix $\mathbf{B}_{S\times K}$ is created by replicating 33 columns for each of the following three columns in their respective order: $[0.77\ 0.64\ 0]'$, $[0.9\ 0\ 0.42]'$ and $[0\ 0.31\ 0.64]'$. Given the current generated return matrix $\mathbf{R}$, the benchmark index is simulated as follows:

$$\mathbf{Y} = \mathbf{R}\mathbf{w} + \nu, \tag{6}$$

where $\mathbf{Y}$ is the $T \times 1$ benchmark return vector, $\mathbf{w}$ is the $K \times 1$ true weight vector for the assets and ν is a $T \times 1$ error matrix drawn from a normal $\sigma \times \mathcal{N}(0,1)$ distribution, with $\sigma = 0.0015$. In detail, the true weight vector is considered as follows: $\mathbf{w} = \frac{1}{\sum_{i=1}^{K} w_i}[0\ 0 \ldots 0\ 0\ 2\ 2 \ldots 2\ 2\ 3\ 3 \ldots 3\ 3]'$. We define the true weights, but also the return matrix $\mathbf{R}$ in such a way that our simulated data exhibits a grouping property to analyze the SLOPE properties. The initial number of groups is equal to three, where all the assets in each group are assigned the same weight.

Like Kremer et al. (2022), we also follow Bogdan et al. (2013) for our simulation and choose the lambda sequence as follows:

$$\lambda_i = \alpha\Phi^{-1}(1 - q_i), \forall i = 1, \ldots, k, \tag{7}$$

where Φ is the standard normal cumulative distribution, $q_i = \frac{i\times\theta}{2k}$ regulates how fast the lambda sequence is decreasing, with $\theta = 0.1$ and α varies on a grid of 12-log spaced points between $10^{-3.5}$ and $10^{-1.7}$. Lastly, each lambda component of the first lambda sequence will be set equal to each other and therefore the SLOPE penalty is equivalent to the LASSO penalty in this case.

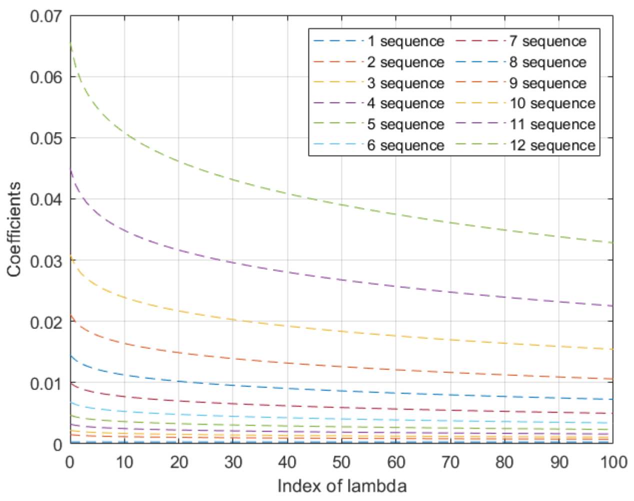


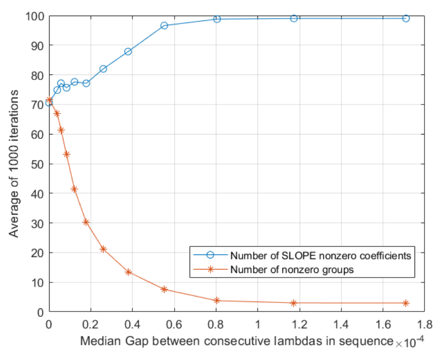
Figure 1: 12 different sequences of lambda parameters

In Figure 1, we can see the 12 different lambda sequences. The 12th lambda sequence represents the sequence with the biggest median gap between consecutive lambda parameters, whereas the first sequence has a median gap of zero. For each of these lambda sequences, we perform 1000 iterations and compute their average number of non-zero groups, non-zero coefficients and also the Mean Squared Error (MSE) and the Mean Squared Prediction Error (MSPE) defined as:

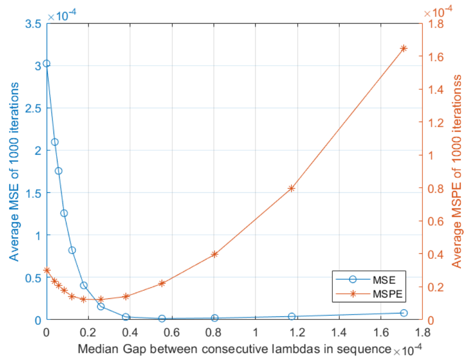
$$MSE(\hat{\mathbf{w}}) = E[||\mathbf{w} - \hat{\mathbf{w}}||_2^2] \tag{8}$$

$$MSE(\hat{\mathbf{w}}) = E[||\mathbf{R}\mathbf{w} - \mathbf{R}\hat{\mathbf{w}}||_2^2]. \tag{9}$$

Panel A of Figure 2 shows for each lambda sequence the number of non-zero coefficients as well as the number of non-zero groups. We observe that for the first sequence, these two coincidence with each other.



(a) Number of non-zero groups and non-zero SLOPE coefficients for each sequence



(b) The Average MSE and MSPE for all 12 lambda sequences

Figure 2: Average number of non-zero coefficients, non-zero groups, MSE and MSPE plotted against the median gap between consecutive lambda parameters in a sequence

As all the lambda parameters are equal, the SLOPE penalty becomes the LASSO penalty resulting in the absence of any grouping. When the median gap between consecutive lambda parameters increases, the number of non-zero groups decreases substantially and converges to the initial number of groups, which is three. This corresponds to Theorem 2 described in Section 2. The SLOPE estimator groups assets together more frequently when the difference between consecutive lambda parameters increases. In addition, we see that the number of non-zero coefficients starts at approximately 70 and as we move to the right of the x-axis, it converges to the total number of simulated assets, which is 99. We initially started with one group that had zero as their true weights. As the magnitude of the lambda sequence increases, SLOPE assigns less weight to the second and third group and increases the estimated weight of the first group to find the optimal solution.

In general, for our simulation, the estimated weight for the third group is the greatest, followed by the estimated weight for the second group, and then the estimated weight for the first group. For each group, the estimated weight is close to the true weight. This is expected given how our optimization problem is defined in (1) and how our benchmark return vector $\mathbf{Y}$ is simulated. Since $\mathbf{Y}$ is multiplied by the true known weights and we want to minimise the tracking error of the given benchmark return, SLOPE puts the estimated weights as closely to the true weights. However, since higher weights are penalized more, it assigns less weight to groups with the highest true weights and more weight to assets in the first group as the median gap between consecutive lambda parameters increases. Consequently, the assets in the first group have a weight slightly greater than zero, which results in no zero elements depicted in Figure 2 Panel A.

Next, in Figure 2 Panel B, we depict the average MSE and average MSPE for each lambda sequence. As the magnitude of the lambda sequence increases, we observe that both the MSE and MSPE decline and find a minimum. After this, both the MSE and MSPE increase again. However, this pattern is weaker for the average MSE. Also, this deviates from the average MSE found in Kremer et al. (2022). We were unable to determine why the average MSE differs from their paper. Nonetheless, when the gap between consecutive lambda parameters increases, the SLOPE penalty becomes stronger and the grouping property gets significant impact on the estimated weights, resulting in greater MSE and MSPE.

3.2 Out-of-sample analysis

To investigate the out-of-sample performance of our tracking portfolio, we compute the out-of-sample excess returns and several out-of-sample statistics. Accordingly, we use a rolling window approach with a window size of $\tau = 750$ suggested by Kremer et al. (2022). The rolling window approach works as follows: at time $t = 750$, we use the first τ observations to get the estimated weight vector $\hat{\mathbf{w}}$ and hold this weight composition for $n = 21$ days to rebalance the tracking portfolio monthly in order to get $M = 101$ out-of-sample observations. Then, we compute the out-of-sample excess return: $\mathbf{Y_{t+1}} - \mathbf{R_{t+1}w_t}$, where $\mathbf{Y_{t+1}}$ is the 21×1 out-of-sample benchmark return vector and $\mathbf{R_{t+1}}$ is a 21×1 constituents return vector. Next, we take the mean to get the average out-of-sample excess return for the next month given the weights at time t.

Again, we follow Kremer et al. (2022) and use the lambda sequence in (7), where we create a grid of 100 linearly spaced starting lambda values for each penalty function by varying α between 10^{-4} and 10^{-2}. In addition, for LASSO, it holds that $\lambda_{\mathbf{LASSO}} = \lambda_1$ such that LASSO has the same lambda parameter as all other penalty functions for only the first asset. Similarly to Kremer et al. (2022), we add the no-short constraint to the optimization problem in (1) for the out-of-sample analysis of our tracking portfolio, as this is typically considered in the IT framework. Since we create 100 linearly spaced α values and thereby creating 100 different lambda sequences, we need to find the optimal tuning parameter that minimizes the optimization problem in (1), given the no-short constraint. Kremer et al. (2022) select a information criterion that is inspired by the Bayesian Information Criteria (BIC) and makes a trade-off between the tracking error volatility and the number of active weights:

$$SC = -2 \times \ln(\frac{\sum_{t=1}^{\tau}(\mathbf{Y_{t+1}} - \mathbf{R_{t+1}\hat{w}_t})^2}{\tau}) + \ln(\tau) \times \sum_{i}^{K} \mathbf{1}(w_i \neq 0), \tag{10}$$

where $\mathbb{1}$ is the indicator function. However, the used criterion in Kremer et al. (2022) is incorrect because the first term is completely dominated by the second term. Also, the notation of indices is incorrect and the current criteria only takes into account the non-zero weights for the last out-of-sample observation per lambda sequence. This seems odd, so we take the mean of the non-zero weights computed for each out-of-sample observation in order to obtain the average non-zero weights. Our slightly other criterion is then displayed as follows:

$$SC = (Mn) \times \ln(\frac{\sum_{t=1}^{M}(\mathbf{Y_{t+1}} - \mathbf{R_{t+1}\hat{w}_t})^2}{Mn}) + \ln(Mn) \times \sum_{t=1}^{M}\sum_{i}^{K} \mathbf{1}(w_i \neq 0). \tag{11}$$

The major difference between the criteria in (10) and (11) is that -2 and τ in their criteria are replaced by Mn, the total number of out-of-sample observations. Consequently, the first term becomes in the same order of magnitude as the second term such that there is actually a trade-off between the tracking error volatility and the number of active weights. It does seems like that Kremer et al. (2022) used the correct criterion for their empirical analysis and that the error in their paper is merely a typo.

After obtaining the optimal lambda sequence corresponding to the lowest BIC, we compute the following out-of-sample test statistics. First, we evaluate the tracking ability of the tracking portfolios by means of the Tracking Error (TE) and Tracking Error Volatility (TEV):

$$TE = \frac{1}{M}\sum_{t=1}^{M}(\mathbf{Y_{t+1}} - \mathbf{R_{t+1}\hat{w}_t}), \tag{12}$$

$$TEV = \frac{1}{M}\sum_{t=1}^{M}(\mathbf{Y_{t+1}} - \mathbf{R_{t+1}\hat{w}_t})^2, \tag{13}$$

where $\mathbf{Y_{t+1}} - \mathbf{R_{t+1}\hat{w}_t}$ is the out-of-sample excess return. Ideally, our tracking portfolio has a TE around zero. A positive TE for a certain tracking portfolio indicates that the benchmark return outperforms the tracking portfolio, whereas a negative TE means that the tracking portfolio outperforms the index. We also calculate the information ratio given by the ratio of the TE to the TEV. A information ratio close to zero is ideal because then the tracking portfolio return closely mimics the benchmark return. Furthermore, we compute the number of active positions (AP) in a portfolio and the turnover (TO) of the replicating portfolio since we are considering sparse and stable clones:

$$AP = \frac{1}{M}\sum_{t=1}^{M}\sum_{i=1}^{K}\mathbb{1}(w_{i,t} \neq 0) \tag{14}$$

$$TO = \frac{1}{M-1}\sum_{t=1}^{M}\sum_{i=1}^{K}|w_{i,t-1} - w_{i,t}|. \tag{15}$$

Here, there is again a small mistake in their notation for their TO statistic. Specifically, we calculate first differences between the estimated portfolio weights thereby losing one observation (i.e. M is replaced by M-1). Moreover, we are interested in the predictive ability of the mimicking portfolio, so we compute the following out-of-sample predictive R^2_{OOS}:

$$R^2_{OOS} = 1 - \frac{\sum_{t=1}^{M}(\mathbf{Y_{t+1}} - \mathbf{R_{t+1}\hat{w}_t})^2}{\sum_{t=1}^{M}\mathbf{Y^2_{t+1}}}, \tag{16}$$

where $-\infty \leq R^2_{OOS} \leq 1$.

Lastly, we use the Diebold-Mariano (DM) test to test for significant differences in the out-of-sample predictive accuracy between two models. The DM test is calculated as follows:

$$DM = \frac{\bar{d}}{\hat{\sigma}_d} \sim \mathcal{N}(0,1), \tag{17}$$

where $d_{t+1} = (\hat{e}^{(1)}_{t+1} - \hat{e}^{(2)}_{t+1})$ and $\hat{e}^{(m)}_{t+1} = \mathbf{Y_{t+1}} - \mathbf{R_{t+1}}\hat{\mathbf{w}}^{(m)}_{\mathbf{t}}$. $\hat{e}^{(m)}_{t+1}$ is the out-of-sample prediction error at time $t+1$, $\hat{\mathbf{w}}^{(m)}_{\mathbf{t}}$ is the estimated weight vector at time t, and $\bar{d}$ and $\hat{\sigma}_d$ represent, respectively, the mean loss differential and Newey-West standard error of the $M \times 1$ loss differential vector d. We reject the null hypothesis of equal forecast accuracy if the mean loss differential is significantly different from zero.

Table 1: Out-of-sample tracking statistics for the S&P indices

	LASSO		SLOPE	
	SP100	*SP500*	*SP100*	*SP500*
Tracking error (%)	0.744	1.302	0.662	1.281
Tracking error volatility (%)	4.910	3.074	4.952	2.900
Information ratio	0.152	0.424	0.134	0.442
Active positions	79	272	81	311
Turnover (%)	0.063	0.080	0.044	0.082
Predicted R^2_{OOS}	0.989	0.994	0.989	0.994

Notes: Reported are the tracking error, tracking error volatility, information ratio, active positions, turnover and predicted R^2_{OOS} for the S&P indices using LASSO and SLOPE. The tracking error and tracking error volatility are annualized.

Table 1 shows the out-of-sample tracking statistics for the S&P100 and S&P500 for different penalty functions. We aim to find a tracking portfolio that is sparse and stable such that it is more cost-effective for investors to create such portfolios. We observe that SLOPE outperforms LASSO for the S&P100 and S&P500 in terms of their annualized tracking error. The annualized tracking error for the S&P500 is also greater than the TE for the S&P100 for both penalties as it is harder to properly track an index with a larger number of constituents. On the other hand, LASSO outperforms SLOPE in terms of their annualized TEV for the S&P100, but only slightly. Moreover, SLOPE creates less sparse portfolios for both indices than LASSO. SLOPE outperforms LASSO in terms of their turnover for the S&P100, while LASSO outperforms SLOPE for the S&P500. Lastly, the R^2_{OOS} are equivalent.

Thus, SLOPE performs the best for the S&P100 creating a sparse portfolio with the lowest turnover while still maintaining equal and even better tracking abilities compared to LASSO. For the S&P500, LASSO seems to outperform SLOPE in terms of their turnovers and active positions, but it has a higher TE and TEV. However, the TE of these two models are not significantly different from each other as determined with the DM test. Both strategies do create a sparse portfolio, but still almost 95% of the constituents is incorporated in the portfolio. For the S&P500, this is significantly better as around 70% of the assets are used.

Comparing the results to Kremer et al. (2022), we observe similar out-of-sample relationships between LASSO and SLOPE, with the exception of the turnover rate. In our study, SLOPE is a cheaper investment strategy than LASSO for the S&P100, while for the S&P500, LASSO has a lower average turnover in comparison with SLOPE. Also, the numerical values for all the out-of-sample statistics are in the same order of magnitude, but still different, especially for the annualized TEV. Possible reasons could be that the initial dataset that has been used to compute the tracking portfolios differs from the dataset used in Kremer et al. (2022) since fewer assets are used in our dataset for both the S&P100 and S&P500. Moreover, the range of α is not specified in Kremer et al. (2022) and the optimal tuning parameter chosen is based on a different criterion that is possibly incorrect. When we use a complete different range for α, the created lambda sequences are totally different, which then causes our SC to choose an optimal lambda sequence from a set of lambda sequences that already differ from those in Kremer et al. (2022). In light of the replicability of the results in Kremer et al. (2022) and the relevance of the lambda components for creating tracking portfolios and its corresponding performance, we are interested in examining the robustness of the SLOPE estimator's output using various forms of lambda sequences.

3.3 Simulation study two-dimensional asset universe

Before that, we first get some intuition using the exact same simulation study in Section 3.1 but with a $I_{2\times 2}$ matrix as $\mathbf{R}$, which indicates a portfolio of two assets that are uncorrelated with each other. For the true weights, we use [0.4 0.6]. Mathematically, the optimization problem in (1) becomes as follows:

$$\begin{aligned} &||\mathbf{Y} - \mathbf{w}||_2^2 + \rho_\lambda(\mathbf{w}) \\ \text{s.t.}\ \ &\mathbf{1}^\intercal \mathbf{w} = \mathbf{1} \\ &\mathbf{w} \geq \mathbf{0}, \end{aligned} \tag{18}$$

where $\rho_\lambda(\mathbf{w}) = \lambda_1$ for LASSO and $\rho_\lambda(\mathbf{w}) = \lambda_1|w_{(1)}| + \lambda_2|w_{(2)}|$ for SLOPE and similar constraints for $\lambda = [\lambda_1, \lambda_2]'$ and $\mathbf{w} = [w_1, w_2]'$ as in (2). Again, we add the no-short constraint to the optimization problem in (1).

Running the simulation, Table 2 shows that the estimated weight vector $\hat{\mathbf{w}}$ converges to the respective elements of the true weights regardless of SLOPE or LASSO. However, there is a slight difference between the estimated weights calculated using SLOPE and LASSO. The SLOPE estimator assigns relatively less weight on $\hat{w}_2$ compared to LASSO when the magnitude of the lambda sequence increases and in absolute sense, $\hat{w}_2$ will always be lower than its corresponding true weight.

Table 2: Asset weights in a two-dimensional universe for all 12 lambda sequences using LASSO and SLOPE

penalty function	assets	1th seq.	2th seq.	3th seq.	4th seq.	5th seq.	6th seq.	7th seq.	8th seq.	9th seq.	10th seq.	11th seq.	12th seq.
LASSO	first	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400	0.400
	second	0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600	0.600
SLOPE	first	0.400	0.400	0.400	0.400	0.400	0.400	0.401	0.401	0.401	0.402	0.402	0.403
	second	0.600	0.600	0.600	0.600	0.600	0.600	0.599	0.599	0.599	0.598	0.598	0.597

This is expected given the aforementioned penalties. LASSO is stuck in the no-short sale area, so the penalty has no effect on the estimated weights. This results in estimated weights that are identical to their true weights. This is not entirely true for SLOPE, as the consecutive lambda parameters differ such that it sets less weight on the most important asset. In this scenario, asset two is assigned a lower weight than its true weight but still close to it.

In general, the estimated weights in a simulation study closely match the actual weights. This holds for every lambda function. However, it deviates slightly as the lambda sequence increases because the penalty becomes stronger. Besides, Table 3 shows the estimated weights for SLOPE when the magnitude of the elements of the lambda sequence are multiplied by a factor of 100. We see that if the penalty becomes strong enough, the SLOPE estimator converges to the equally weighted portfolio. Thus, assets that are uncorrelated with each other still can be grouped together using SLOPE. The estimated weights using LASSO will not differ and therefore still be equal to the estimated weights in Table 2.

Table 3: Asset weights in a two-dimensional universe for all 12 lambda sequences using LASSO and SLOPE

penalty function	assets	1th seq.	2th seq.	3th seq.	4th seq.	5th seq.	6th seq.	7th seq.	8th seq.	9th seq.	10th seq.	11th seq.	12th seq.
SLOPE	first	0.405	0.407	0.411	0.416	0.423	0.433	0.448	0.470	0.500	0.500	0.500	0.500
	second	0.595	0.593	0.589	0.584	0.577	0.567	0.552	0.530	0.500	0.500	0.500	0.500

Notes: The 12 lambda sequences are multiplied by a factor of 100.

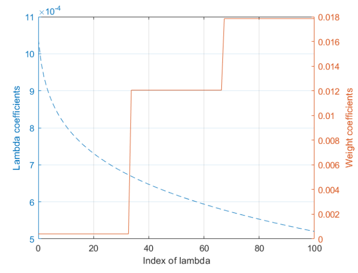
Now, changing the form of the lambda sequence could also affect the results of SLOPE. In the most extreme case, where we set $\lambda_1 = 10^{-2}\Phi^{-1}(1-q_1)$ and $\lambda_2 = 0$, the two assets are not grouped together. A possible explanation could be that the difference between the partial correlations of the two assets with the benchmark index is larger than the difference in the two lambda parameters. Referring back to Theorem 2, both the magnitude and the form of a lambda sequence should have impact on the grouping property of SLOPE. In this simple scenario, it seems that the magnitude of lambda has a greater effect.

4 Extension

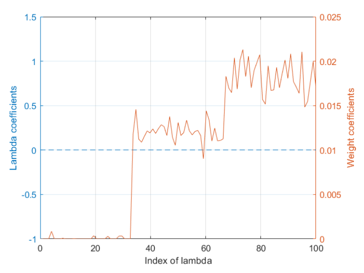
This section reports a simulation study using different lambda sequences. Moreover, we compute the out-of-sample tracking statistics for two new lambda functions using empirical data. We use daily return observations of the S&P100 and S&P500 from 31 December 2004 to 29 January 2016 ($T = 2890$ observations) gathered by Datastream. Lastly, we also discuss the impact of excluding the no-short selling constraint, given an imposed budget constraint for LASSO and SLOPE.

4.1 Different lambda sequences

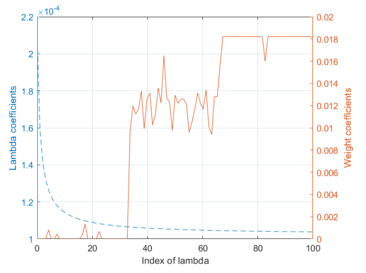
From our intuition, we represent eight different lambda functions that are displayed with a blue line in Figure 3. Additionally, we display the estimated weights computed using the respective lambda function. The x-axis depicts the number of assets, the left y-axis the order of magnitude for the lambda components and the right y-axis the estimated weight coefficients. We again run the exact simulation study that is introduced in Section 3.1. The eight different lambda functions are named as follows: LASSO, Inverse Gaussian, Indicator, Linear, Power, Power2, Jump and Jump2. All lambda functions are analysed within the same order of magnitude and the estimated weights are determined using the 12th sequence (i.e. highest median gap between consecutive lambda parameters) for each lambda function. The Indicator function takes lambda values equal to one for the first half of the assets and zero otherwise. The Linear function is a lambda sequence that decreases linearly. The Power function declines rapidly for the first few assets and thereafter decreases very slowly. The Power2 function is the exact opposite of the Power function. Next, the Jump function is a combination of the Linear- and Indicator function. It divides the sequence into three equal parts. The lambda parameters are equal to each other for the first part, declines linearly for the second part and stays the same for the last part. Lastly, the Jump2 function is created in the same way as the Jump function. However, it declines linearly for the first part, stays the same for the second part and again declines linearly for the last part.



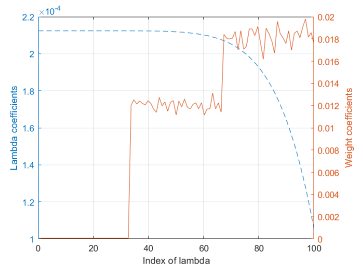
(a) Inverse Gaussian



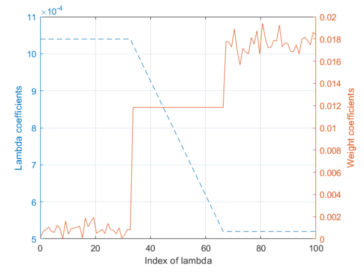
(b) LASSO



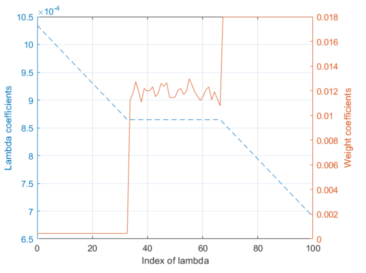
(c) Power



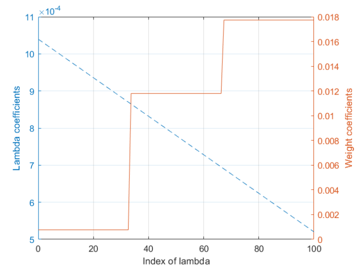
(d) Power2



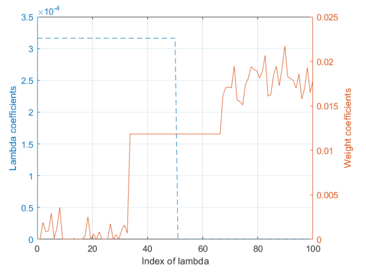
(e) Jump



(f) Jump2



(g) Linear



(h) Indicator

Figure 3: eight different lambda functions and the corresponding estimated coefficient weights

4.1.1 Investor portfolio construction

Only the Inverse Gaussian and Linear functions can explicitly determine the initial grouping structure of our return matrix $\mathbf{R}$, as depicted in Figure 3. For the two power functions we observe that when the lambda function is flat, assets are not grouped together. When the function goes down, the estimated weights are set equal to each other, thus proposing some grouping of assets. Note that in our simulation study, the first group always gets the lowest estimated weights followed by the second group, and lastly the third group, which gets assigned the largest estimated weight. Therefore, we see a returning stepwise pattern for all the estimated coefficient weights.

In Figure 3 panel C, we see a horizontal line for the estimated weights for the third group. Since the highest lambda parameters correspond with the highest estimated weights and the gap between consecutive lambda parameters for the first few lambda components is very big, SLOPE thus groups these assets together. A similar explanation can be done for the two Jump functions. For example, in Figure 3 Panel E, we see that assets in the second group are all assigned different weights. This is due to equal corresponding lambda parameters. Therefore, SLOPE does not group these assets together even though their correlation to the benchmark index are approximately similar. No grouping of assets is observed for LASSO, as all weights are distinct. Only the Indicator function displays rather odd results for the estimated weights. Initially, we expect almost no grouping of assets as the gap between most of the consecutive lambda components is equal to zero. However, the orange line in Panel H shows that the estimated weights in the second group are set equal to each other. Thus, for most of the lambda functions, SLOPE shows its grouping ability. But it also shows that how assets are grouped is dependent on which lambda function is chosen.

As a result, from the perspective of an investor, we can now select a lambda sequence based on our prior knowledge of the index or on the desired portfolio construction properties. For example, if we want that the least important assets are assigned the same weight and that the rest of the coefficient weights for the remaining assets can be varying, then the Power2 function fits best. An investor may find this function or even a slightly modified version in which the Power2 function decreases even earlier to be very appealing. As the least important assets for the benchmark index explain relatively less, we could set their weights to equal, whereas we would allow the weights of the most important assets to vary, given their greater explanatory power. Prior knowledge about the benchmark index is thus needed. If an investor desires that the most important and least important assets have the same weight, but the estimated weights for the assets in between can be different, the Jump2 function is appropriate for this type of portfolio construction. Furthermore, the Inverse Gaussian or Linear function can be used to create a tracking portfolio with the smallest number of groups possible. Lastly, the LASSO function fits best if you do not wish to group your assets at all, such that each asset is assigned a unique weight.

4.1.2 Changing initial parameters

In Table 4, the simulation shows that only the Linear and Inverse Gaussian function converge to the initial number of groups, which is three as the magnitude of the lambda sequence increases. None of the other sequences converge to the initial number of groups. It is reasonable for the LASSO function as it has no grouping property. Surprisingly, the number of non-zero groups even increase for the Jump and Indicator function as the magnitude of the lambda sequence increases. Changes to the initial true weights influence the estimated weights for each lambda function. Making two of the three groups their true weight similar to each other causes the number of non-zero groups to converge to two for the Linear and Inverse Gaussian function as the magnitude of the lambdas increases. Using the other lambda functions results in significantly less asset grouping. However, on average the estimated weights do divide the assets into two groups, with each group's assets having the same weight on average.

Table 4: Number of non-zero groups for the eight lambda functions using the twelve lambda sequences of different magnitude

eight lambda functions	1th seq.	2th seq.	3th seq.	4th seq.	5th seq.	6th seq.	7th seq.	8th seq.	9th seq.	10th seq.	11th seq.	12th seq.
Inverse Gaussian	69	66	61	53	41	30	20	13	7	3	3	3
LASSO	71	71	71	71	71	71	71	71	71	71	71	70
Power	71	71	70	70	68	67	64	61	56	51	45	41
Power2	71	71	71	71	71	71	71	69	67	66	66	65
Jump	62	53	44	41	41	42	44	46	49	53	58	65
Jump2	71	70	69	66	60	51	40	35	35	35	35	35
Linear	70	69	64	55	42	25	9	3	3	3	3	3
Indicator	60	55	49	44	41	41	42	43	45	48	52	57

Furthermore, we already observed in a simplified scenario that the magnitude of the lambda components impacts the SLOPE estimator. Increasing the lambda parameters by a factor of 10 naturally results in fewer non-zero groups for all eight different lambda functions. Now, the number of non-zero groups first increases for the Indicator and Jump function as the magnitude of the lambda sequence increases, but eventually declines just as the rest of the lambda functions. Scaling all lambda functions by a factor of 100, the Indicator, Jump, Jump2, Linear and Inverse Gaussian function put all weights equal to each other, producing the equally estimated weighted portfolio. As the penalty becomes more severe, it pushes the weights equal to each other, completely ignoring the correlation between assets. As the median gap between consecutive lambda parameters in the LASSO function is zero, the number of non-zero groups remains unchanged regardless of scaling.

Moreover, through simulation, we show that the form of the sequence does have impact on whether or not certain assets are assigned the same weight and therefore grouped together. For example, as the magnitude of the lambda parameters for the Jump2 function increases, the number of non-zero groups converges to exactly 35, as shown in Table 4. This number of non-zero groups is not arbitrary. Assets within the first or third group get equal weights as the lambda parameters increases. However, according to the Jump2 function, the second group of assets have different weights. Looking at the form of the Jump2 function, we see that the lambda parameters are equal to each other for the second part. Therefore, it yields identical results to LASSO, producing no groups. Since each groups consists of 33 assets, this would result in 33 non-zero groups plus the first and third group, resulting in 35 non-zero groups. Similar logic can be applied to the Jump function, yielding a total of 67 non-zero groups as the magnitude of the lambda parameters increases.

In general, the form of the lambda sequence has impact on how assets are grouped together. Especially when the sequence is steep, it has a higher probability to group the corresponding assets of these lambda components together or when the sequence is rather flat, it has a higher probability to assign different weights to assets with approximately similar partial correlation to the benchmark index. Nonetheless, the magnitude has a greater influence on the grouping structure of the SLOPE estimator, as it also groups uncorrelated assets together.

Currently, we only run our newly created lambda functions in a simulation. Therefore we extend Table 1 with the Linear and Jump function using empirical data, as the Linear function shows similar results as the Inverse Gaussian function in terms of their ability to create non-zero groups. On the other hand, the Jump function may not result in more sparse portfolios than the SLOPE and the Linear function. However, it could have better tracking abilities. We will not elaborate on the remaining six functions, as the aforementioned functions are more interesting.

4.2 Sensitivity

Table 5 shows the out-of-sample tracking statistics for LASSO, SLOPE and the new included lambda functions Linear and Jump. Compared to LASSO and SLOPE, the Linear and Jump functions have lower tracking errors for the S&P100. The Linear function has the lowest TE for the S&P500. In terms of the TEV, the Jump function outperforms the other functions for the S&P100. SLOPE and the Linear function have the lowest TEV for the S&P500. We do see that the Linear and Jump function have more active positions than LASSO and SLOPE for both indices.

Table 5: Out-of-sample tracking statistics for the S&P indices

	LASSO		SLOPE		Linear		Jump	
	SP100	*SP500*	*SP100*	*SP500*	*SP100*	*SP500*	*SP100*	*SP500*
Tracking error (%)	0.744	1.302	0.662	1.281	0.607	1.262	0.632	1.327
Tracking error volatility (%)	4.910	3.074	4.952	2.900	4.889	2.900	4.848	2.994
Information ratio	0.152	0.424	0.134	0.442	0.124	0.436	0.130	0.443
Active positions	79	272	81	311	82	340	82	351
Turnover (%)	0.063	0.080	0.044	0.082	0.040	0.079	0.041	0.081
Predicted R^2_{OOS}	0.989	0.993	0.989	0.994	0.989	0.994	0.989	0.994

Notes: Reported are the tracking error, tracking error volatility, information ratio, active positions, turnover and predicted R^2_{OOS} for the S&P indices using LASSO, SLOPE, the Linear and Jump function. The tracking error and tracking error volatility are still annualized.

Consequently, the created tracking portfolios are less sparse and thus more costly. The average turnover and predictive R^2_{OOS} of the two new lambda functions are identical to SLOPE's values. Thus, in terms of the tracking ability, the Linear function outperforms the other lambda functions. However, the tracking abilities for the SLOPE and Linear function are approximately equal due to similar grouping behaviour found in Section 4.1.2. Indeed, using the DM test, we observe that the tracking errors between SLOPE and Linear for the S&P indices are not significantly different from each other. Yet, from an investor perspective, the Linear function could create tracking portfolios that are a little less sparse, but has better tracking abilities.

Table 6: DM statistics calculated by comparing the squared prediction errors of the SLOPE function to the other three functions

DM statistic	*S&P100*	*S&P500*
LASSO	0.215	-0.532
Linear	0.654	1.094
Jump	0.439	-0.884

Table 6 reports all the DM statistics of the different models using SLOPE as the benchmark model. The mean loss differential with SLOPE is not significantly different from zero not only for the Jump function but also for the other functions. In addition, the squared prediction errors for SLOPE are lower for the S&P500 than for the LASSO and Jump function, since the DM statistic is negative. We examine that the out-of-sample results and thereby also the estimated weights are dependent on the form of the lambda sequence. Though, the four lambda functions are not significantly different from each other in terms of their out-of-sample tracking abilities.

Lastly, if we increase the magnitude of the four different functions, the penalty becomes too strong such that the SLOPE, Linear and the Jump function create a portfolio that approaches the equally weighted portfolio. Likewise, the TE and TEV are higher while the average turnover becomes smaller (See Appendix A, Table 8). The difference between the values of these out-of-sample statistics when the magnitude is altered is also greater than when the form of the lambda sequence is altered. The LASSO function on the other hand, is not affected by an increase or decrease in the lambda components. Thus, as was also shown in Section 4.1.2, the SLOPE estimator is more dependent on the magnitude of the lambda components than the form of the lambda sequence.

4.3 No-short constraint

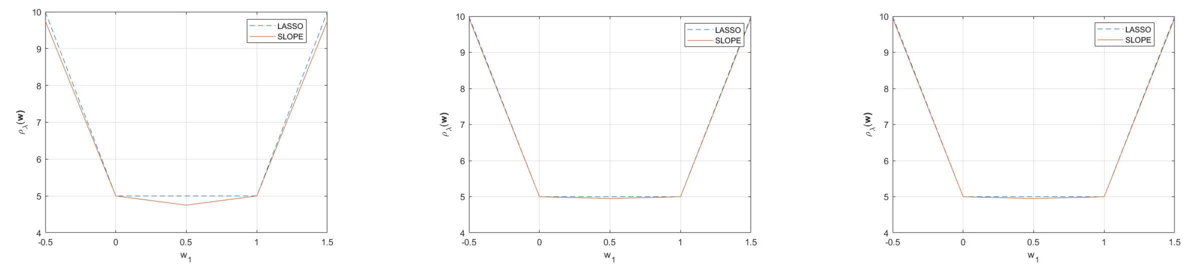
This section elaborates on the no-short selling constraint that leads to a problem for the LASSO penalty and potentially for the SLOPE penalty. We look at a two-dimensional asset world to see how the LASSO and SLOPE penalty functions behave, given the no-short selling constraint. Furthermore, We use the exact simulation study in Section 3.1 to analyze the SLOPE estimator using lambda sequences that closely correspond to a lambda sequence used for the LASSO penalty.

In a two-dimensional asset universe, the LASSO and SLOPE penalty are defined as follows:

$$\rho_\lambda(w) = \lambda_{LASSO}(|w_1| + |w_2|) \tag{19}$$

$$\rho_\lambda(w) = \lambda_1|w_{(1)}| + \lambda_2|w_{(2)}|, \tag{20}$$

where $\lambda_1 \geq \lambda_2 \geq 0$ and $|w_i|$ is the ith largest entry in the estimated weight vector. Solving the optimization problem in (1) and excluding the no-short selling constraint (i.e. $w_i \geq 0$, for $i = 1, 2$), the value of the LASSO penalty $\rho_\lambda(\mathbf{w})$ differs when varying the two weights. However, as stated by DeMiguel et al. (2009), the LASSO penalty becomes stuck in the no-short selling area because the penalty is always equal to λ_{LASSO}, regardless of the weights. Figure 4 displays the LASSO and SLOPE penalty, given the imposed budget constraint.



(a) $\lambda_1 = 5$, $\lambda_2 = 2.5$ (b) $\lambda_1 = 5$, $\lambda_2 = 4.5$ (c) $\lambda_1 = 5$, $\lambda_2 = 4.9$

Figure 4: LASSO and SLOPE penalty functions in a two-dimensional asset world

On the x-axis, the weights for w_1 are displayed, and on the y-axis, the value for the certain penalty. Panel A of Figure 4 shows visually that LASSO is stuck in the no-short sale area. Namely, when $0 \leq w_1 \leq 1$, the LASSO penalty becomes a horizontal line. Consequently, the value of the penalty remains constant regardless of varying the weights. Considering the SLOPE penalty, optimizing problem (1) and also excluding the no-short selling constraint, the value of the SLOPE penalty will differ for each combination of the two weights. Adding the no-short selling constraint to the problem leads initially to no problems. The two asset weights are still penalized differently. Thus varying these weights lead to different outcomes for $\rho_\lambda(\mathbf{w})$. In panel A of Figure 4, it shows that the minimum for the SLOPE penalty is obtained for the equally weighted portfolio.

However, when the difference between λ_1 and λ_2 decreases, the SLOPE penalty converges to the LASSO penalty, as is shown in Panel B and C of Figure 4. When the difference of the two lambdas is only 0.5, we see that the SLOPE penalty almost becomes a horizontal line for the long-only area, meaning no impact to the problem in (18). Narrowing the gap between the two lambdas even further, Panel C of Figure 4 then shows that the two lines almost coincidence with each other. Therefore, the SLOPE penalty becomes approximately equal to the LASSO penalty.

Solving problem (1) and including the no-short selling constraint thus leads to a problem for the LASSO penalty since it will be always equal to λ_{LASSO}. Therefore, the estimated weights are solely dependent on the return matrix $\mathbf{R}$ and benchmark return vector $\mathbf{Y}$. Nonetheless, imposing the no-short-selling constraint has no effect on the results for any of the penalty functions if all estimated weights are positive. For the SLOPE function, it does not necessarily imply that the weights are only dependent on the return matrix $\mathbf{R}$ and the benchmark return vector $\mathbf{Y}$ when a no-short selling constraint is included.

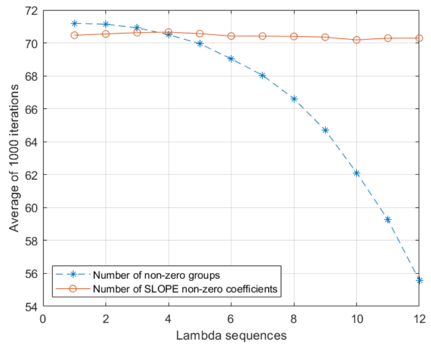
To show this, we propose a similar simulation as in Section 3.1. The only difference is the lambda sequence. In detail, we define two different lambda sequences:

$$\lambda_{conv1} = \alpha[0.60, 0.30, \ldots, 0.30] \tag{21}$$

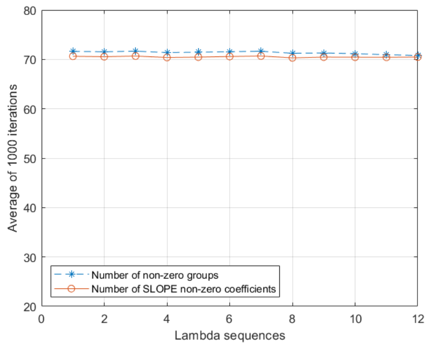
$$\lambda_{conv2} = \alpha[0.31, 0.30, \ldots, 0.30], \tag{22}$$

where α varies on a grid of 12 log-spaced points between $10^{-3.5}$ and $10^{-1.7}$. Therefore, we have 12 lambda sequences for both λ_{conv1} and λ_{conv2}. The only difference between the two lambda sequences is the first lambda value, with the first lambda value of λ_{conv2} being much closer to the other lambda components than the first lambda value of λ_{conv1}. Consequently, λ_{conv2} is almost a horizontal line such that the SLOPE penalty converges to the LASSO penalty, whereas the SLOPE penalty using λ_{conv1} leads to relative less similarity with the LASSO penalty.

Figure 5 shows the simulation results using these two new lambda sequences. It displays the number of non-zero groups for each of the 12 lambda sequences. In panel A of Figure 5 we can see that the



(a) Number of non-zero groups using λ_{conv1}



(b) Number of non-zero groups using λ_{conv2}

Figure 5: Average number of non-zero groups and non-zero SLOPE coefficients using λ_{conv1} and λ_{conv2}

number of non-zero groups decreases as the median gap between the lambda parameters increases. The number of non-zero groups is still higher than the initial number of groups, which was 3. In spite of the no-short-selling constraint and when nearly all lambda components are identical, SLOPE is still capable of grouping certain assets together. On average, SLOPE is even able to identify the initial grouping structure as the assets are assigned the same average weight per group. Initially, we would expect that the number of non-zero groups would not decline so much since almost all parameters are equal. However, we observe that the steepness between consecutive lambda parameters has a greater impact for the grouping property of SLOPE. As long as one lambda component of the lambda sequence differs substantially, SLOPE still holds its grouping properties to some extent.

Panel B of Figure 5 displays the number of non-zero groups and non-zero SLOPE coefficients using λ_{conv2}. Here, we see that using λ_{conv2}, the number of non-zero groups stays the same as the magnitude of the lambda parameters increase. Since λ_{conv2} is a substantially more horizontal line in comparison with λ_{conv1}, we observe that the steepness of the lambda sequence has drastic impact on the grouping property of SLOPE. For both sequences, it holds that the number of non-zero groups and number of non-zero SLOPE coefficients are not exactly equal as not all lambda parameters are exactly equivalent.

As discussed earlier, changing the magnitude of all the 12 lambda sequences would decrease the number of non-zero groups even more if we use λ_{conv1}. However, SLOPE does not create the equally weighted portfolio when all lambda components are multiplied by a factor of 100. If we use λ_{conv2}, which causes the SLOPE penalty to be more closely related to a LASSO penalty than λ_{conv1}, then changing the magnitude would have no effect on the number of non-zero groups. Thus, we conclude that if we use λ_{conv2}, which converges to the LASSO penalty, for our SLOPE penalty, then the SLOPE estimator loses its theoretical properties. Moreover, the steepness and magnitude of the lambda sequence play an important role for the SLOPE performance.

5 Conclusion

The first aim of this paper is to answer the following research question:

RQ1 : *Are the weight positions in the tracking portfolios obtained from the SLOPE estimator of Kremer et al. (2022) sensitive to different lambda sequences?*

We find through simulation- and empirical analysis that the steepness of the lambda sequence and the magnitude of the lambda parameters have influence on the weight positions of the created tracking portfolio. When the lambda sequence is steep, it has a higher probability to group the corresponding assets of these lambda components together. SLOPE can push assets into groups based on a predefined form of the lambda sequence. Lastly, the magnitude of the lambda sequence has a greater effect on the SLOPE estimator. When the penalty is strong enough, an equally weighted portfolio is created in which all assets are grouped together.

Next, this paper shows potential problems of the SLOPE estimator when it converges to the LASSO estimator, which is a special case of SLOPE by answering the following research question:

RQ2 : *Does the SLOPE estimator also suffer in the no short-sale area given an imposed budget constraint as it converges to LASSO?*

It is shown that as the SLOPE penalty converges to the LASSO penalty, the SLOPE estimator loses its grouping properties and therefore also becomes stuck in the no-short sale area.

As suggestions for further research, we scale the lambda sequences with fairly arbitrary factors. Further research can be done to find the optimal magnitude of a certain lambda sequence such that SLOPE still performs well and therefore minimizes the out-of-sample statistics. This could be accomplished using a linearly spaced scalar and a similar empirical analysis. Moreover, further work can also be done in finding the difference between consecutive lambda parameters such that the SLOPE estimator creates the equally weighted portfolio regardless of other parameters. Indirectly, the difference between two consecutive lambda parameters increases as the magnitude of the components increase. A suggestion is to run a simulation in a two-dimensional universe in which the partial correlation of the two assets with the benchmark index are exactly the opposite with lambda sequences of different steepness. One can then find the threshold where SLOPE still groups these two assets together, given the optimal magnitude of the lambda sequences.

6 Bibliography

References

M. Bogdan, E. v. d. Berg, W. Su, and E. Candes. Statistical estimation and testing via the sorted l1 norm. *arXiv preprint arXiv:1310.1969*, 2013.

M. Bogdan, E. Van Den Berg, C. Sabatti, W. Su, and E. J. Candès. Slope—adaptive variable selection via convex optimization. *The annals of applied statistics*, 9(3):1103, 2015.

H. D. Bondell and B. J. Reich. Simultaneous regression shrinkage, variable selection, and supervised clustering of predictors with oscar. *Biometrics*, 64(1):115–123, 2008.

V. DeMiguel, L. Garlappi, F. J. Nogales, and R. Uppal. A generalized approach to portfolio optimization: Improving performance by constraining portfolio norms. *Management science*, 55(5):798–812, 2009.

M. Giuzio. Genetic algorithm versus classical methods in sparse index tracking. *Decisions in Economics and Finance*, 40(1):243–256, 2017.

P. J. Kremer, S. Lee, M. Bogdan, and S. Paterlini. Sparse portfolio selection via the sorted ℓ1-norm. *Journal of Banking & Finance*, 110:105687, 2020.

P. J. Kremer, D. Brzyski, M. Bogdan, and S. Paterlini. Sparse index clones via the sorted ℓ1-norm. *Quantitative Finance*, 22(2):349–366, 2022.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Appendix A Tables

Table 7: Summary statistics of the S&P indices.

Index	k	$\hat{\mu}$ (%)	$\hat{\sigma}$ (%)	Median (%)	Min.	Max.	Skewness	Kurtosis
S&P100	86	0.014	1.204	0.030	−9.186	10.655	−0.271	11.542
S&P500	409	0.016	1.243	0.038	−9.470	10.957	−0.329	11.312

Table 8: Out-of-sample tracking statistics for the S&P indices

	LASSO		SLOPE		Linear		Jump	
	SP100	*SP500*	*SP100*	*SP500*	*SP100*	*SP500*	*SP100*	*SP500*
Tracking error (%)	0.579	1.240	-1.344	-1.222	-1.735	-1.644	-1.832	-1.724
Tracking error volatility (%)	5.003	2.956	9.436	9.376	9.655	12.205	9.816	12.388
Active positions	79	276	85	370	85	396	85	398
Turnover (%)	0.124	0.160	0.013	0.022	0.009	0.012	0.008	0.011

Notes: Reported are the tracking error, tracking error volatility, active positions and turnover for the S&P indices using LASSO, SLOPE, the Linear and Jump function. The tracking error and tracking error volatility are annualized. The lambda functions are scaled by a factor of 100.

Appendix B Description MATLAB code

In this guide we give a short explanation how we obtained our results using the MATLAB code that is found in "thesisCode".

First we explain how to get the results in Section 3. To obtain Figures 1 and 2, run the script "SimulationStudy99assets.m". For the out-of-sample statistics for LASSO and SLOPE, run the script "EmpiricalOutOfSample.m". Furthermore, Table 2 and 3 are computed by running the script "SimulationStudy2dimensional.m".

Second, we explain how to obtain the results in Section 4. Figure 4 and Table 4 are obtained by running script "SimulationStudy99assets.m". Moreover, the out-of-sample statistics for the two new lambda functions are again obtained by running script "EmpiricalOutOfSample.m". Figure 4 and 5 are created by running the script "SimulationStudy2dimensional.m" and "SimulationStudy99assets.m" respectively.

Lastly, Table 7 from Appendix A is obtained using the "BenchmarkReturnSP100SP500" Excel sheet. We run "EmpiricalOutOfSample.m" to get Table 8.