Synthetic Control Method in Asset Pricing

Jesus Villota Miranda[†]

⟨ [†]CEMFI, Calle Casado del Alisal, 5, 28014 Madrid, Spain ⟩

⟨ Email: jesus.villota@cemfi.edu.es ⟩

This version: 17th December 2024

Abstract

This paper develops a novel framework for applying the Synthetic Control Method (SCM) to asset pricing and portfolio management. We extend the traditional SCM methodology by incorporating financial market features, including short-selling capabilities and regularization techniques to manage transaction costs and portfolio concentration. The framework is further enhanced through machine learning approaches that capture nonlinear relationships between financial instruments. We demonstrate the method's versatility through various applications, including statistical arbitrage, hedging of complex securities, ETF replication, and risk analysis. Our methodology provides practitioners with a systematic approach to construct synthetic portfolios that closely track target instruments while maintaining implementation feasibility through controlled trading costs and portfolio turnover.

JEL Codes: C14, C45, G11, G12, G13

Keywords: Synthetic Control Method; Machine Learning; Asset Pricing; Portfolio Management; Statistical Arbitrage; Risk Management; ETF Replication; Nonlinear Methods

SCM in No Arbitrage framework

Structure of the Paper

1. **Theoretical Framework**
   - **Models**
     - Basic Linear Model (SC is a traded asset)
     - Extension: Linear Model with Regularization (SC is a traded asset)
     - Nonlinear Model with Regularization (SC is NOT a traded asset)
   - **Econometric & Asymptotic Analysis**
2. **Practical Applications**
   - **SCM for Statistical Arbitrage**
     - We only use the SC to generate the trading signals ⇒ we don't need the SC to be a traded asset ⇒ we can use the nonlinear model
     - Compare the profitability of the strategy when using: SC, DeepSC, OU, DeepSA, where:
       * SC = Linear Regularized Synthetic Control [Villota]
       * DeepSC = Nonlinear Regularized Synthetic Control [Villota]
       * OU = Ostern Uhlenbeck process [Avellaneda & Lee]
       * DeepSA = Deep Learning in Statistical Arbitrage [Replicated SA method from Guijarro-Ordoñez, Pelger, Zanotti]
       * DeepSA-Rank = Deep Statistical Arbitrage in Rank Space [Li & Papanicolau]
   - **SCM for Hedging**
     - We need a traded SC ⇒ use Linear Regularized SC

1. Methodology

The Synthetic Control Method (SCM) is a powerful statistical technique initially developed for causal inference in comparative case studies. In asset pricing, SCM can be adapted to construct a synthetic portfolio that closely replicates the return dynamics of a target financial instrument by optimally weighting a combination of the returns of other instruments.

- If we want to trade the synthetic asset, say, for hedging or other purposes, we should focus on linear combinations, as nonlinear combinations of assets cannot be traded (that is, we cannot buy Apple^2 or $\sqrt{\text{Apple} \cdot \text{Nvidia}}/\text{Meta}$)
- If we simply want to generate a synthetic return series in order tom compare R_0 to $\hat{R}_0$, look at the deviations: $\delta = |R_0 - \hat{R}_0|$ and generate trading signals: $Z = \frac{\delta - \mu(\delta)}{\sigma(\delta)}$. Then, define a set of thresholds $\{c_{\text{open-long}}, c_{\text{close-long}}, c_{\text{open-short}}, c_{\text{close-short}}\}$ such that a trading signal is generated based on the region where Z lies.

1.1 Traditional SCM

Consider a universe of financial instruments $\mathcal{I} := \{0, 1, \ldots, N\}$ and discrete time periods $\mathcal{T} := \{1, \ldots, T\}$. Let R_{it} denote the returns of instrument i at time t and take instrument 0 to be our target. Our goal is to mimick the returns of instrument 0 with a linear combination of returns from a donor pool $\mathcal{J} := \{1, \ldots J\}$

$$R^*_{0t} = \sum_{j \in \mathcal{J}} w_j R_{jt} \ ,$$

where the weights $\{w_j\}_{j \in \mathcal{J}}$ are chosen to minimize the tracking error over a training period $\mathcal{T}_{tr} \subset \mathcal{T}$

$$\min \sum_{t \in \mathcal{T}_{tr}} (R_{0t} - \sum_{j \in \mathcal{J}} w_j R_{jt})^2 \qquad \text{s.t.} \quad \sum_{j \in \mathcal{J}} w_j = 1 \ .$$

Different from traditional SCM, we don't impose a positivity constraint on the weights, as in finance, it makes sense to consider negative weights (it just implies short selling).

1.2 Generalized Linear SCM with Regularization

Define $T_{tr} = |\mathcal{T}_{tr}|$ as the number of periods in the training sample, $\mathbf{R}_0 \in \mathbb{R}^{T_{tr}}$ as the vector with the time series of target returns, and $\mathbf{R} \in \mathbb{R}^{T_{tr} \times J}$ as the panel with the returns of the donor pool over the training sample. Then, a more sophisticated SMC is:

$$\begin{aligned} \min_{\mathbf{w}} \quad & \|\mathbf{R}_0 - \mathbf{R}\mathbf{w}\| + \lambda \mathcal{R}(\mathbf{w}) \\ \text{s.t.} \quad & \mathbf{1}^\top \mathbf{w} = 1 \end{aligned}$$

where $\|\cdot\|$ denotes some norm, and $\mathcal{R}(\mathbf{w})$ is a regularization term that can take various forms depending on the desired properties of the solution. The hyperparameter $\lambda \geq 0$ is selected through cross-validation and it controls the strength of the regularization. When $\lambda = 0$, we recover the basic SCM formulation. As λ increases, the solution becomes more regularized, trading off tracking error for more desirable weight properties. Common choices of $\mathcal{R}(\cdot)$ include:

- **Lasso Regularization (L1 Penalty)**: $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_{i=1}^{N} |w_i|$ which promotes sparsity in $\mathbf{w}$, potentially setting some weights to zero. It is specially useful when the number of donor assets N is large.
- **Ridge Regularization (L2 Penalty)**: $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_2^2 = \sum_{i=1}^{N} w_i^2$ which shrinks weights towards zero but does not enforce exact zeros. It stabilizes the solution when predictors are highly correlated.
- **Elastic Net Regularization**: Combines L1 and L2 penalties: $\mathcal{R}(\mathbf{w}) = \alpha\|\mathbf{w}\|_1 + (1-\alpha)\|\mathbf{w}\|_2^2$ where the hyperparameter $\alpha \in [0, 1]$ balances between Lasso and Ridge penalties.

1.3 Nonlinear Synthetic Control via Machine Learning

To capture nonlinearities, we extend SCM by incorporating machine learning (ML) techniques. Instead of modeling the target returns as a linear combination of donor assets, we model the relationship using a nonlinear function $f_{\boldsymbol{\theta}} : \mathbb{R}^J \to \mathbb{R}$ parameterized by $\boldsymbol{\theta} \in \Theta$

$$R_{0t}^* = f_{\boldsymbol{\theta}}(\mathbf{R}_t) \quad t \in \mathcal{T}_{tr} \ ,$$

where $\mathbf{R}_t \in \mathbb{R}^J$ is the vector of donor asset returns at time t. Our objective is to learn the function $f_{\boldsymbol{\theta}}$ that maps the donor pool returns to the target asset returns. This can be achieved by minimizing a suitable loss function over the training period $\mathcal{T}_{tr}$

$$\min_{\boldsymbol{\theta}} \quad \frac{1}{T_{tr}} \sum_{t \in \mathcal{T}_{tr}} L\left(R_{0t}, f_{\boldsymbol{\theta}}(\mathbf{R}_t)\right) + \lambda \mathcal{R}(\boldsymbol{\theta}),$$

where $L(\cdot, \cdot)$ is a loss function (e.g., mean squared error), $\mathcal{R}(\boldsymbol{\theta})$ is a regularization term to prevent overfitting, and $\lambda \geq 0$ is a hyperparameter controlling the strength of regularization.

1.3.1 Machine Learning Models

Several machine learning models can be employed to capture nonlinear relationships:

- **Neural Networks (NNs)**: NNs are flexible function approximators capable of capturing complex nonlinear patterns. A feedforward neural network with L layers can be defined recursively as:

$$\begin{aligned} \mathbf{h}^{(0)} &= \mathbf{R}_t, \\ \mathbf{h}^{(l)} &= \sigma^{(l)}\left(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right), \quad l = 1, \ldots, L-1, \\ f_{\boldsymbol{\theta}}(\mathbf{R}_t) &= \mathbf{W}^{(L)}\mathbf{h}^{(L-1)} + b^{(L)}, \end{aligned}$$

  where $\sigma^{(l)}(\cdot)$ is an activation function (e.g., ReLU, sigmoid), $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are weight matrices and bias vectors, and $\boldsymbol{\theta} = \{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\}_{l=1}^{L}$.

- **Kernel Methods**: Kernel regression models, such as Support Vector Regression (SVR), can capture nonlinearities by implicitly mapping inputs into a high-dimensional feature space via a kernel function $K(\mathbf{R}_t, \mathbf{R}_s)$.

- **Tree-Based Models**: Models like Random Forests and Gradient Boosting Machines (e.g., XGBoost) use ensembles of decision trees to model nonlinear interactions between variables.

- **Gaussian Processes**: Probabilistic models that define a distribution over functions and can provide uncertainty estimates along with predictions.

1.3.2 Training and Validation

To train the ML models, we split the data into training and validation sets. The training set ($\mathcal{T}_{tr}$) is used to fit the model parameters $\boldsymbol{\theta}$, while the validation set ($\mathcal{T}_{val}$) is used to tune hyperparameters and prevent overfitting.

- **Cross-Validation**: Techniques like k-fold cross-validation can be employed to assess model performance and robustness.

- **Regularization**: Regularization techniques such as weight decay (L2 regularization), dropout (for neural networks), and early stopping help mitigate overfitting.

- **Hyperparameter Tuning**: Hyperparameters, including network architecture, learning rate, and regularization strength, are optimized using validation performance metrics.

1.3.3 Model Evaluation

The performance of the ML-based SCM is evaluated using appropriate metrics:

- **Mean Squared Error (MSE)**: Measures the average squared difference between the predicted and actual returns.
- **Mean Absolute Error (MAE)**: Measures the average absolute difference, providing robustness to outliers.
- **Out-of-Sample Testing**: Evaluate the model on unseen data to assess generalization performance.

1.3.4 Interpretability

To understand the nonlinear relationships learned by the model:

- **SHAP Values**: Calculate Shapley values to quantify the contribution of each donor instrument
$$\phi_j = \sum_{S \subseteq \mathcal{J} \setminus \{j\}} \frac{|S|!(|\mathcal{J}| - |S| - 1)!}{|\mathcal{J}|!} [f_S(\{j\}) - f_S(\emptyset)]$$
- **Partial Dependence Plots**: Visualize how changes in one donor instrument affect the synthetic return while averaging over other instruments
$$\text{PDP}_j(x) = \mathbb{E}_{R_{-j}}[f_{\boldsymbol{\theta}}(R_j = x, R_{-j})]$$

The ML-based SCM can capture complex nonlinear relationships while maintaining interpretability through these analysis tools. This approach is particularly valuable when the linear SCM fails to adequately replicate the target instrument's returns.

2. Practical Applications

2.1 SCM for Statistical Arbitrage

Statistical arbitrage strategies can be developed using SCM by identifying and exploiting temporary price discrepancies between a target asset and its synthetic replica. The methodology is particularly suitable for this application because:

- **Pairs Trading Extension**: While traditional pairs trading focuses on two similar assets, SCM allows for constructing a more sophisticated synthetic counterpart using multiple assets, potentially capturing the target's characteristics more accurately.
- **Trading Signals**: Deviations between the target asset and its synthetic replica can generate trading signals. When the spread between them exceeds a predetermined threshold (e.g., 2 standard deviations), we can simultaneously:
  - Long (short) the target asset if it's trading below (above) its synthetic value
  - Short (long) the synthetic portfolio with the optimal weights
- **Risk Management**: The regularization terms help control portfolio turnover and transaction costs, making the strategy more practical to implement. Specifically:
  - L1 regularization promotes sparse portfolios, reducing transaction costs
  - L2 regularization stabilizes weights over time, reducing turnover

2.1.1 Methodology

Let R_0 be the return series of a target asset and let $\hat{R}_0$ be the synthetic return series generated by taking a nonlinear combination of the return series of a donor pool of assets.

We simply want to generate a synthetic return series in order to compare R_0 to $\hat{R}_0$

Algorithm

1. Look at the deviations: $\delta = |R_0 - \hat{R}_0|$
2. Generate trading signals: $Z = \frac{\delta - \mu(\delta)}{\sigma(\delta)}$
3. Define a set of thresholds $\{c_{\text{open-long}}, c_{\text{close-long}}, c_{\text{open-short}}, c_{\text{close-short}}\}$ such that a trading signal is generated based on the region where Z lies.
   - Trading Signal:
     - Open Short if $Z \geq c_{\text{open-short}}$
     - Close Short if $Z \leq c_{\text{close-short}}$
     - Open Long if $Z \leq c_{\text{open-long}}$
     - Close Long if $Z \geq c_{\text{close-long}}$

2.2 SCM for Hedging

SCM provides a systematic approach to constructing hedging portfolios for complex financial instruments or illiquid assets:

- **Options Hedging**: For exotic options or structured products without liquid market hedges, SCM can construct synthetic hedging portfolios using more liquid instruments:
  - Target: Returns of the complex option
  - Donor pool: Returns of vanilla options, the underlying, and related liquid securities
- **Illiquid Asset Hedging**: For assets that trade infrequently or have high transaction costs:
  - Target: Returns of the illiquid asset
  - Donor pool: Returns of liquid assets with similar risk exposures

2.3 SCM for ETF Replication

The method can be applied to create cost-effective alternatives to existing ETFs or to develop new investment vehicles:

- **ETF Shadowing**: Replicate the performance of expensive ETFs using a smaller set of liquid components:
  - Target: ETF returns
  - Donor pool: Major constituent stocks or related liquid instruments
- **Custom Index Tracking**: Create bespoke indices with specific characteristics:
  - Reduced tracking error through optimal weighting
  - Lower implementation costs through regularization-induced sparsity

2.4 SCM for Risk Analysis

The methodology can enhance various aspects of risk management and analysis:

- **Factor Exposure Analysis**: Decompose an asset's returns into exposures to various factors:
  - Target: Asset returns

  - Donor pool: Factor-mimicking portfolios

- **Stress Testing**: Estimate how an asset might perform in scenarios where historical data is limited:
  - Construct synthetic versions using assets with longer histories
  - Use the synthetic portfolio to simulate performance in different market conditions
- **Liquidity Risk Assessment**: Evaluate the replicability of an asset's returns using liquid alternatives:
  - Higher tracking error suggests greater liquidity risk
  - Regularization parameters can be tuned to reflect transaction cost constraints

3. Appendix

3.1 Asymptotic Framework

3.1.1 Assumptions

We begin by establishing the necessary assumptions for our asymptotic analysis:

Assumption 1 (Data Generating Process)**.** *For each asset i and time t, returns follow:*

$$R_{it} = \mu_i(F_t) + \epsilon_{it} \tag{1}$$

where F_t is a vector of common factors, $\mu_i(\cdot)$ is a continuous function, and ϵ_{it} satisfies:

(a) $E[\epsilon_{it}|F_t] = 0$

(b) $E[\epsilon_{it}\epsilon_{jt}|F_t] = 0$ *for* $i \neq j$

(c) $E[\epsilon_{it}\epsilon_{is}|F_t, F_s] = 0$ *for* $t \neq s$

Assumption 2 (Mixing Conditions)**.** *The sequence $\{(R_{it}, F_t)\}_{t=1}^{T}$ is strictly stationary and α-mixing with mixing coefficients $\alpha(h)$ satisfying:*

$$\sum_{h=1}^{\infty} h^2 \alpha(h)^{\delta/(2+\delta)} < \infty \tag{2}$$

for some $\delta > 0$.

Assumption 3 (Moment Conditions)**.** *For all i and t:*

(a) $E|R_{it}|^{4+\delta} < \infty$

(b) $E|\epsilon_{it}|^{4+\delta} < \infty$

(c) $\sup_t E\|F_t\|^{4+\delta} < \infty$

3.2 Consistency Theory

3.2.1 Weight Convergence

Let w_T^* denote the optimal weights estimated using T observations. We establish:

Theorem 1 (Weight Consistency)**.** *Under Assumptions 1-3, as $T \to \infty$:*

$$\|w_T^* - w^0\| \xrightarrow{p} 0 \tag{3}$$

where w^0 represents the population optimal weights.

Proof. The proof proceeds in three steps:

1. First, we show that the objective function converges uniformly:

$$\sup_{w \in \mathcal{W}} |Q_T(w) - Q(w)| \xrightarrow{p} 0 \tag{4}$$

where

$$Q_T(w) = \frac{1}{T}\sum_{t=1}^{T}(R_{it} - \sum_{j=1}^{J} w_j R_{jt})^2 \tag{5}$$

$$Q(w) = E[(R_{it} - \sum_{j=1}^{J} w_j R_{jt})^2] \tag{6}$$

2. Using the mixing conditions, we apply a uniform law of large numbers:

$$\|Q_T(w) - Q(w)\|_\infty = O_p(T^{-1/2}\log T) \tag{7}$$

3. Finally, we establish identification of w^0 through the positive definiteness of the second moment matrix of returns. □

3.2.2 Rate of Convergence

We establish the convergence rate under additional regularity conditions:

Theorem 2 (Convergence Rate)**.** *Under Assumptions 1-3 and suitable regularity conditions:*

$$\sqrt{T}(w_T^* - w^0) \xrightarrow{d} N(0, V) \tag{8}$$

where

$$V = H^{-1}\Sigma H^{-1} \tag{9}$$

with

$$H = E[\nabla^2 Q(w^0)] \tag{10}$$

$$\Sigma = \lim_{T\to\infty} Var(\sqrt{T}\nabla Q_T(w^0)) \tag{11}$$

3.3 Inference Procedures

3.3.1 Asymptotic Distribution

For the synthetic return estimator:

Theorem 3 (Asymptotic Normality)**.** *Under stated assumptions:*

$$\sqrt{T}(R_{it}^S - R_{it}^*) \xrightarrow{d} N(0, \Omega) \tag{12}$$

where

$$\Omega = R_{jt}' V R_{jt} + \sigma_\epsilon^2 \tag{13}$$

3.3.2 Hypothesis Testing Framework

For testing the accuracy of synthetic control matches:

$$H_0 : R_{it}^S = R_{it}^* \quad \text{vs} \quad H_1 : R_{it}^S \neq R_{it}^* \tag{14}$$

Test statistic:

$$\tau_T = T(R_{it}^S - R_{it}^*)' \hat{\Omega}^{-1} (R_{it}^S - R_{it}^*) \tag{15}$$

Under H_0:

$$\tau_T \xrightarrow{d} \chi^2(k) \tag{16}$$

3.4 Finite Sample Properties

3.4.1 Bias Analysis

The finite sample bias of the synthetic control estimator is:

$$E[R_{it}^S - R_{it}^*] = B_T + O(T^{-1}) \tag{17}$$

where

$$B_T = -\frac{1}{2T} tr(H^{-1}\Sigma) + O(T^{-3/2}) \tag{18}$$

3.4.2 Higher-Order Properties

We derive the Edgeworth expansion:

$$P(\sqrt{T}(R_{it}^S - R_{it}^*) \leq x) = \Phi(x) + T^{-1/2} p_1(x)\phi(x) + O(T^{-1}) \tag{19}$$

where $p_1(x)$ is a polynomial depending on the third and fourth moments.

3.5 Robust Inference

3.5.1 HAC Estimation

For robust variance estimation:

$$\hat{\Omega} = \sum_{|h|<m_T} k(h/m_T)\hat{\Gamma}(h) \tag{20}$$

where

$$\hat{\Gamma}(h) = \frac{1}{T}\sum_{t=|h|+1}^{T} \hat{u}_t \hat{u}'_{t-|h|} \tag{21}$$

with $k(\cdot)$ being a kernel function and m_T the bandwidth parameter.

3.5.2 Bootstrap Procedures

We establish the validity of the following bootstrap procedure:

1. Generate bootstrap samples:
$$R^{*b}_{it} = R^{S}_{it} + \epsilon^{*b}_{it} \tag{22}$$

2. Compute bootstrap weights:
$$w^{*b}_T = \arg\min_{w\in\mathcal{W}} \sum_{t=1}^{T} (R^{*b}_{it} - \sum_{j=1}^{J} w_j R_{jt})^2 \tag{23}$$

3. Bootstrap distribution:
$$\sqrt{T}(w^{*b}_T - w^{*}_T) \xrightarrow{d} N(0, V) \tag{24}$$

3.6 Time-Varying Parameter Extensions

3.6.1 Local Asymptotic Framework

For time-varying parameters:

$$w_t = w_0 + h_T \beta(t/T) \tag{25}$$

where $h_T \to 0$ as $T \to \infty$.

Local linear estimator:

$$\hat{w}_t = \arg\min_{w,\beta} \sum_{s=1}^{T} K_h(t-s)(R_{is} - \sum_{j=1}^{J}(w_j + \beta_j(t-s))R_{js})^2 \tag{26}$$

3.6.2 Uniform Inference

We establish uniform confidence bands:

$$P(\sup_{t\in[0,1]} |\hat{w}_t - w_t| \leq c_\alpha \sqrt{\log T/Th}) \to 1-\alpha \tag{27}$$

where c_α is the critical value obtained from the distribution of the supremum of a Gaussian process.

3.7 Implementation Details

For simplicity, consider a weighted L2 norm $\|\cdot\|_{\mathbf{W}}$ defined by a positive definite matrix $\mathbf{W} \in \mathbb{R}^{T_{tr}\times T_{tr}}$. Common choices include the identity matrix $\mathbf{W} = \mathbf{I}$ (standard L2 norm) or the inverse of the sample covariance matrix $\mathbf{W} = \hat{\mathbf{\Sigma}}^{-1}$ (Mahalanobis norm), which accounts for the correlation structure in the returns. The optimization problem can be solved through the method of Lagrange multipliers.

$$\mathcal{L}(\mathbf{w},\mu) = (\mathbf{R}_0 - \mathbf{R}\mathbf{w})^\top \mathbf{W}(\mathbf{R}_0 - \mathbf{R}\mathbf{w}) + \lambda\mathcal{R}(\mathbf{w}) - \mu(\mathbf{1}^\top\mathbf{w} - 1)$$

For differentiable regularization functions $\mathcal{R}(\cdot) \in C$ the gradient is

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = -2\mathbf{R}'\mathbf{W}(\mathbf{R}_0 - \mathbf{R}\mathbf{w}) + \lambda\nabla\mathcal{R}(\mathbf{w}) - \mu\mathbf{1} = \mathbf{0}\ .$$

With Ridge regularization $\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_2^2 = \mathbf{w}^\top\mathbf{w}$ we have $\nabla\mathcal{R}(\mathbf{w}) = 2\mathbf{w}$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}} = -2\mathbf{R}'\mathbf{W}\left(\mathbf{R}_0 - \mathbf{R}\mathbf{w}\right) + 2\lambda\mathbf{w} - \mu\mathbf{1} = \mathbf{0}$$

Simplify:

$$2\left(\mathbf{R}'\mathbf{W}\mathbf{R} + \lambda\mathbf{I}\right)\mathbf{w} - 2\mathbf{R}'\mathbf{W}\mathbf{R}_0 - \mu\mathbf{1} = \mathbf{0}$$

Let $\mathbf{M} = \mathbf{R}'\mathbf{W}\mathbf{R} + \lambda\mathbf{I}$ and $\mathbf{b} = \mathbf{R}'\mathbf{W}\mathbf{R}_0$. Then:

$$\mathbf{M}\mathbf{w} - \mathbf{b} + \frac{\mu}{2}\mathbf{1} = \mathbf{0}$$

From the constraint:

$$\mathbf{1}'\mathbf{w} = 1$$

Stacking the equations:

$$\begin{bmatrix} \mathbf{M} & \frac{1}{2}\mathbf{1} \\ \mathbf{1}' & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mu \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ 1 \end{bmatrix}$$

The resulting linear system can be solved efficiently using standard numerical methods. However, care must be taken as the matrix $\mathbf{M}$ may be ill-conditioned, especially when assets in the donor pool are highly correlated. In such cases, increasing λ can help stabilize the solution.

For the Lasso case ($\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_1$), the optimization problem becomes non-differentiable and requires specialized algorithms such as proximal gradient descent or coordinate descent methods.

3.8 Cross-Validation

The choice of λ is crucial for the performance of the synthetic portfolio. We employ k-fold cross-validation over the training period to select an optimal value. The training period $\mathcal{T}_{tr}$ is divided into k consecutive blocks. For each candidate λ and each fold, we:

1. Estimate weights using $k-1$ blocks
2. Compute out-of-sample tracking error on the held-out block

The final λ is chosen to minimize the average out-of-sample tracking error across folds:

$$\lambda^* = \arg\min_{\lambda} \frac{1}{k} \sum_{\ell=1}^{k} \mathrm{TE}_\ell(\lambda)$$

where $\mathrm{TE}_\ell(\lambda)$ is the tracking error on the i-th validation fold using regularization parameter λ.

3.9 Rolling Window Estimation

In financial markets, the relationships between assets are dynamic and can change over time due to evolving economic conditions, market sentiments, and structural shifts. To capture these time-varying relationships, we implement a rolling window estimation approach for the synthetic control weights $\mathbf{w}_t$. The optimization problem at time t is formulated as:

$$\mathbf{w}_t^* = \underset{\{w_j\}_{j\in\mathcal{J}}}{\arg\min} \sum_{\tau=t-h}^{t-1} v_\tau \left(R_{0\tau} - \sum_{j\in\mathcal{J}} w_j R_{j\tau} \right)^2 + \lambda \mathcal{R}(\mathbf{w}_t) \quad \text{s.t.} \sum_{j\in\mathcal{J}} w_j = 1 \tag{28}$$

for a weighting function $v_s(\cdot)$ and some estimation window h. The weighting function v_τ determines how observations within the window contribute to the estimation:

- **Uniform Weights**: $v_\tau = 1$ for all τ, giving equal importance to all observations in the window.

- **Exponential Weights**: $v_\tau = \exp\left(-\gamma(t-\tau)\right)$, where $\gamma > 0$ controls the rate at which older observations are discounted.
- **Linear Decay**: $v_\tau = \frac{\tau-(t-h)}{h}$, linearly decreasing the weight of older observations.

The choice depends on the desired responsiveness of the model to recent data. In order to avoid overly frequent changes in weughts which would lead to higher transaction costs, we could incorporate a penalty for weight changes or directly include transaction costs in the optimization.

Recursive Least Squares An alternative to explicitly solving the optimization problem in each window is to use recursive least squares (RLS), which updates the weight estimates incrementally:

$$\mathbf{w}_t = \mathbf{w}_{t-1} + \mathbf{K}_t \left(R_{0,t-1} - \mathbf{R}'_{t-1}\mathbf{w}_{t-1}\right),$$

where $\mathbf{K}_t$ is the gain matrix computed based on the covariance of the predictor variables. RLS is efficient and well-suited for online updating.

3.10 Donor Pool

The donor pool $\mathcal{J}$ could be defined in various ways. A lax way to define it would be simply as $\mathcal{J} := \mathcal{S} \setminus \{0\}$ and let regularization select optimally from the highest possible set of donors. Alternatively, we could define it through a hierarchical screening process:

$$\mathcal{J} = \{j : S_j \geq \bar{S} \cap d(X_i, X_j) \leq \delta \cap \rho_{ij} \geq \bar{\rho}\}$$

where:

- S_j represents firm size (market capitalization), with $\bar{S}$ as a minimum threshold
- $d(X_i, X_j)$ is a distance metric between firm characteristics vectors
- ρ_{ij} is the historical return correlation, with $\bar{\rho}$ as a minimum threshold

The characteristics vector X_i includes:

$$X_i = [\text{Size}_i, \text{B/M}_i, \text{Momentum}_i, \text{Volatility}_i, \text{Industry}_i]$$

Dynamically Updated Donor Pool

To maintain pool relevance, we implement a rolling update mechanism:

$$\mathcal{J}_t = \mathcal{J}_{t-1} \cup \mathcal{A}_t \setminus \mathcal{D}_t$$

where $\mathcal{A}_t$ and $\mathcal{D}_t$ represent additions and deletions based on the screening criteria at time t.