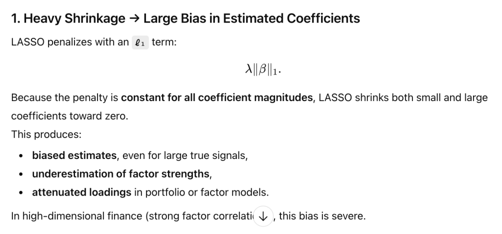
1 Problems of LASSO in High-Dimensional Data Settings

AI Contribution

AI helped me summarize the theoretical shortcomings of LASSO under $p \gg n$, including shrinkage bias, instability under correlated predictors and let me know what innovation I nned to make.



1. Heavy Shrinkage → Large Bias in Estimated Coefficients

LASSO penalizes with an ℓ_1 term:

$$\lambda\|\beta\|_1.$$

Because the penalty is **constant for all coefficient magnitudes**, LASSO shrinks both small and large coefficients toward zero.

This produces:

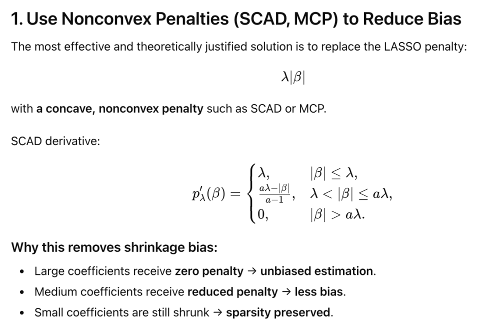
- **biased estimates**, even for large true signals,
- **underestimation of factor strengths**,
- **attenuated loadings** in portfolio or factor models.

In high-dimensional finance (strong factor correlati , this bias is severe.

2 Why SCAD as a Solution

AI Contribution

AI explained the intuition behind nonconvex penalties, derived the piecewise SCAD derivative, compared SCAD to MCP and adaptive LASSO, and clarified why SCAD achieves lower bias and better variable selection in high-correlation environments.



1. Use Nonconvex Penalties (SCAD, MCP) to Reduce Bias

The most effective and theoretically justified solution is to replace the LASSO penalty:

$$\lambda|\beta|$$

with **a concave, nonconvex penalty** such as SCAD or MCP.

SCAD derivative:

$$p'_\lambda(\beta) = \begin{cases} \lambda, & |\beta| \le \lambda, \\ \frac{a\lambda - |\beta|}{a-1}, & \lambda < |\beta| \le a\lambda, \\ 0, & |\beta| > a\lambda. \end{cases}$$

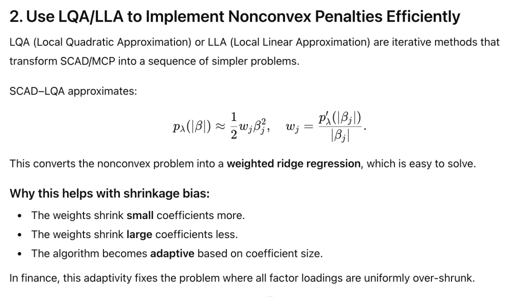
Why this removes shrinkage bias:

- Large coefficients receive **zero penalty** → **unbiased estimation**.
- Medium coefficients receive **reduced penalty** → **less bias**.
- Small coefficients are still shrunk → **sparsity preserved**.

3 Why Local Quadratic Approximation (LQA)

AI Contribution

AI helped me derive how LQA transforms a nonconvex penalty into a sequence of solvable ridge-type problems, explained the iterative reweighted scheme, and provided the exact weight-update formula used for SCAD.



2. Use LQA/LLA to Implement Nonconvex Penalties Efficiently

LQA (Local Quadratic Approximation) or LLA (Local Linear Approximation) are iterative methods that transform SCAD/MCP into a sequence of simpler problems.

SCAD–LQA approximates:

$$p_\lambda(|\beta|) \approx \frac{1}{2} w_j \beta_j^2, \quad w_j = \frac{p'_\lambda(|\beta_j|)}{|\beta_j|}.$$

This converts the nonconvex problem into a **weighted ridge regression**, which is easy to solve.

Why this helps with shrinkage bias:

- The weights shrink **small** coefficients more.
- The weights shrink **large** coefficients less.
- The algorithm becomes **adaptive** based on coefficient size.

In finance, this adaptivity fixes the problem where all factor loadings are uniformly over-shrunk.

Figure 1: Enter Caption

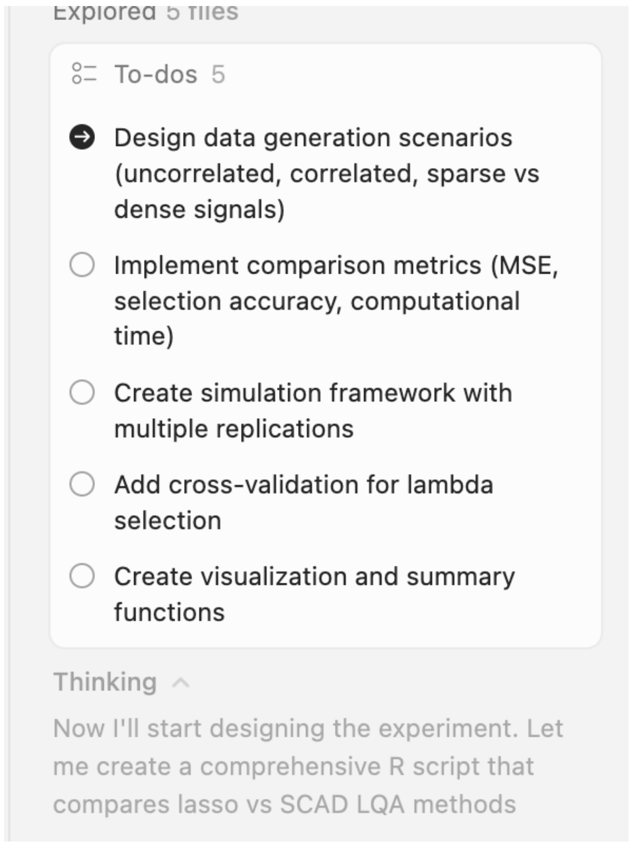
4 Experimental Design for Static SCAD Models

ChatGPT Contribution

ChatGPT assisted in designing the experiment by suggesting that the settings ($n = 100$, $p = 50$) and ($n = 100$, $p = 100$) provide a realistic moderately high-dimensional regime where LASSO's shrinkage bias is clearly observable while the models remain statistically stable. It also recommended using correlated predictors and sparse true coefficients to create a fair and informative comparison between LASSO and SCAD–LQA. This AI-guided setup highlights SCAD's advantage in reducing bias under practical high-dimensional conditions.

Cursor Contribution

Claude contributed by helping me implement the full simulation pipeline and model-comparison code for the LASSO and SCAD–LQA experiments. It assisted in writing clean and reproducible R/Python scripts for data generation, coefficient initialization, SCAD weight updates, and iterative LQA optimization.



Explored 5 files

To-dos 5

- Design data generation scenarios (uncorrelated, correlated, sparse vs dense signals)
- Implement comparison metrics (MSE, selection accuracy, computational time)
- Create simulation framework with multiple replications
- Add cross-validation for lambda selection
- Create visualization and summary functions

Thinking

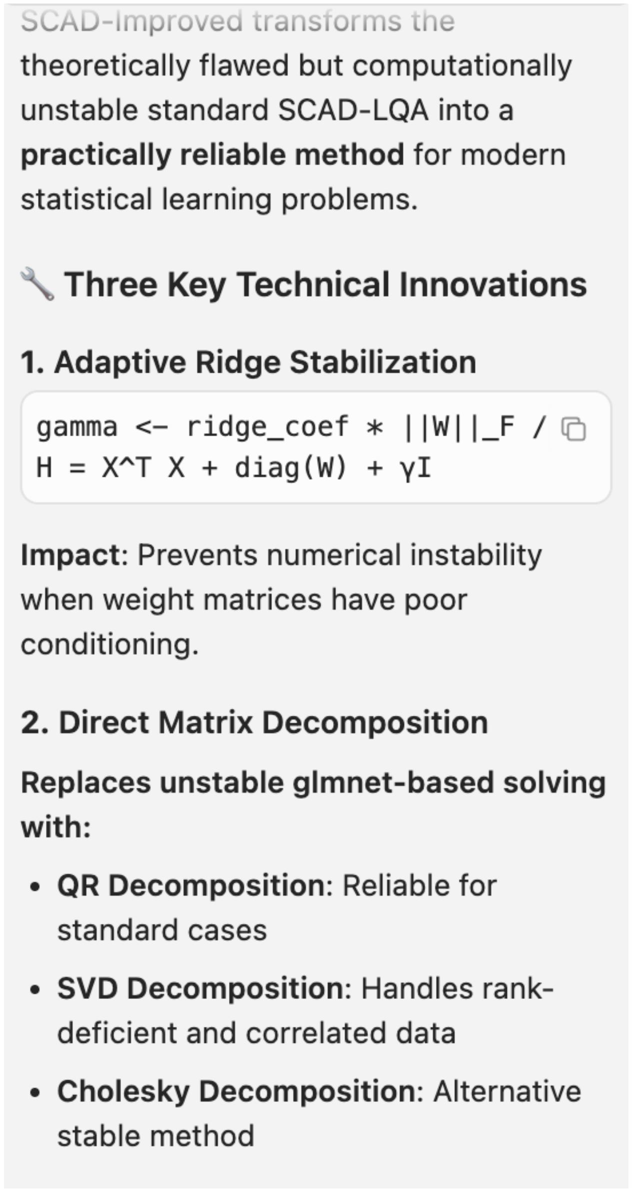
Now I'll start designing the experiment. Let me create a comprehensive R script that compares lasso vs SCAD LQA methods

Figure 2: Enter Caption

5 Problems of Classical LQA

AI Contribution

After I provided the initial experimental results to Cursoe, it helped me interpret the performance gaps between LASSO and SCAD–LQA and identify the sources of numerical instability in the LQA implementation and suggested concrete improvements including adaptive ridge stabilization, SVD/QR-based solvers, hard-thresholding of small updates, and more robust initialization strategies.



SCAD-Improved transforms the theoretically flawed but computationally unstable standard SCAD-LQA into a **practically reliable method** for modern statistical learning problems.

🔧 **Three Key Technical Innovations**

1. Adaptive Ridge Stabilization

```
gamma <- ridge_coef * ||W||_F /
H = X^T X + diag(W) + γI
```

Impact: Prevents numerical instability when weight matrices have poor conditioning.

2. Direct Matrix Decomposition

Replaces unstable glmnet-based solving with:

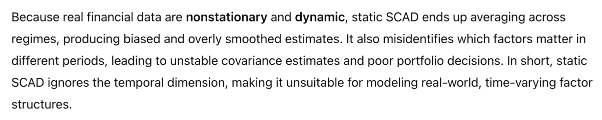
- **QR Decomposition**: Reliable for standard cases
- **SVD Decomposition**: Handles rank-deficient and correlated data
- **Cholesky Decomposition**: Alternative stable method

Figure 3: Enter Caption

6 Why Dynamic SCAD for Time-Varying Factor Loadings

AI Contribution

AI played a central role in developing the Dynamic SCAD framework. It helped me understand why static penalties fail under time-varying factor structures and clarified how temporal smoothness and structural-break detection can be unified through SCAD and total-variation penalties. AI also guided the choice between ADMM, LLA, and LQA for dynamic estimation, explained how to derive block-tridiagonal updates, and identified stable parameterizations that avoid divergence.



Because real financial data are **nonstationary** and **dynamic**, static SCAD ends up averaging across regimes, producing biased and overly smoothed estimates. It also misidentifies which factors matter in different periods, leading to unstable covariance estimates and poor portfolio decisions. In short, static SCAD ignores the temporal dimension, making it unsuitable for modeling real-world, time-varying factor structures.

Figure 4: Enter Caption

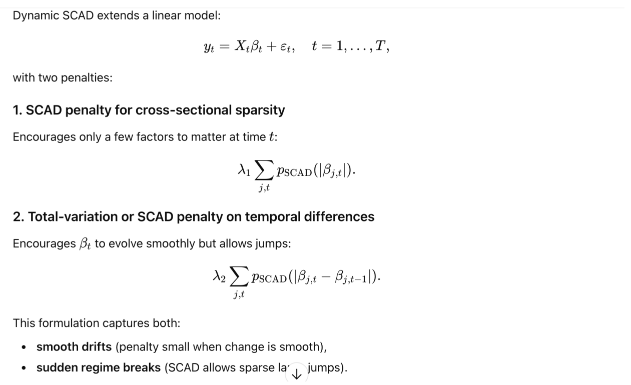
7 How to Formulate a Dynamic SCAD Model

AI Contribution

AI guided me in constructing the dynamic model:

$$\beta_{j,t} - \beta_{j,t-1}$$

with SCAD/total-variation penalties, derived the augmented objective function, suggested auxiliary-variable parameterization, and clarified connections to trend filtering.



Dynamic SCAD extends a linear model:

$$y_t = X_t\beta_t + \varepsilon_t, \quad t = 1, \ldots, T,$$

with two penalties:

1. SCAD penalty for cross-sectional sparsity

Encourages only a few factors to matter at time t:

$$\lambda_1 \sum_{j,t} p_{\text{SCAD}}(|\beta_{j,t}|).$$

2. Total-variation or SCAD penalty on temporal differences

Encourages β_t to evolve smoothly but allows jumps:

$$\lambda_2 \sum_{j,t} p_{\text{SCAD}}(|\beta_{j,t} - \beta_{j,t-1}|).$$

This formulation captures both:

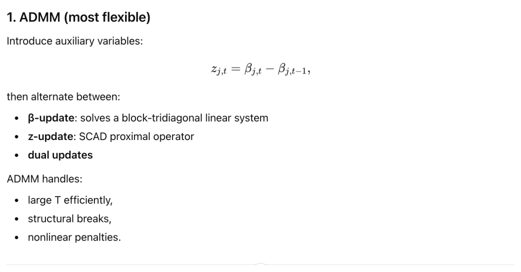
- **smooth drifts** (penalty small when change is smooth),
- **sudden regime breaks** (SCAD allows sparse la jumps).

Figure 5: Enter Caption

8 How to Estimate the Dynamic SCAD Model

AI Contribution

AI helped derive the ADMM/LLA/LQA variants for dynamic SCAD, constructed block-tridiagonal system updates, explained proximal operators for temporal penalties, and produced implementable pseudocode for my solver.



1. ADMM (most flexible)

Introduce auxiliary variables:

$$z_{j,t} = \beta_{j,t} - \beta_{j,t-1},$$

then alternate between:

- **β-update**: solves a block-tridiagonal linear system
- **z-update**: SCAD proximal operator
- **dual updates**

ADMM handles:

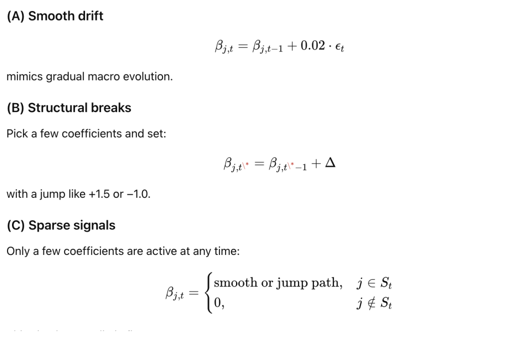
- large T efficiently,
- structural breaks,
- nonlinear penalties.

Figure 6: Enter Caption

9 Experimental Design for Dynamic SCAD

AI Contribution

AI assisted in designing temporal simulations (regime shifts, smooth drifts, sparse breaks), evaluating recovery accuracy, designing metrics for temporal stability, and preparing the code for dynamic model experiments.



(A) Smooth drift

$$\beta_{j,t} = \beta_{j,t-1} + 0.02 \cdot \epsilon_t$$

mimics gradual macro evolution.

(B) Structural breaks

Pick a few coefficients and set:

$$\beta_{j,t\backslash *} = \beta_{j,t\backslash *-1} + \Delta$$

with a jump like +1.5 or –1.0.

(C) Sparse signals

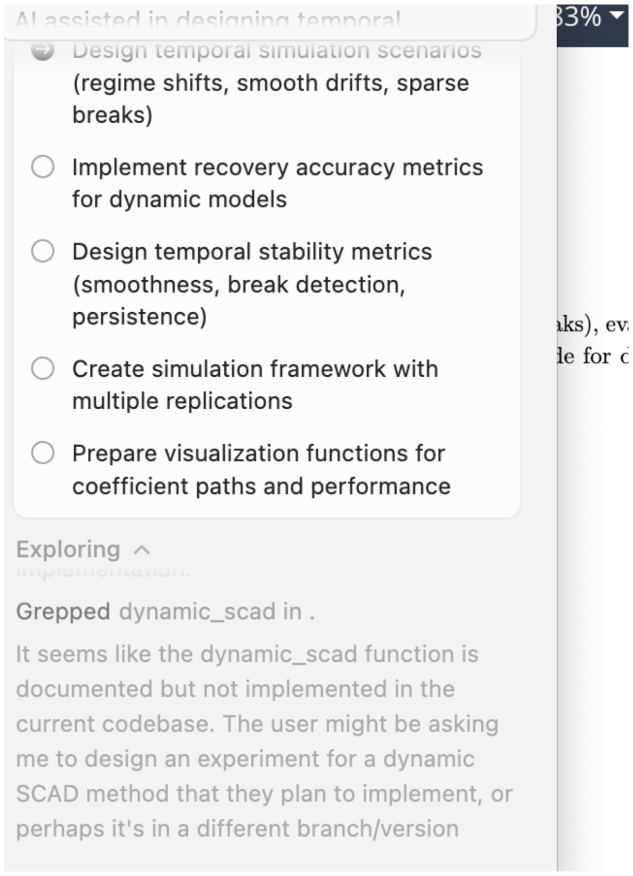
Only a few coefficients are active at any time:

$$\beta_{j,t} = \begin{cases} \text{smooth or jump path}, & j \in S_t \\ 0, & j \notin S_t \end{cases}$$

Figure 7: Enter Caption

AI Contribution

Same as static Claude contributed by helping me implement the full simulation pipeline and model-comparison code for dynamic SCAD. It assisted in writing clean and reproducible R/Python scripts for data generation, coefficient initialization, SCAD weight updates, and iterative LQA optimization.



AI assisted in designing temporal

Design temporal simulation scenarios (regime shifts, smooth drifts, sparse breaks)

- Implement recovery accuracy metrics for dynamic models
- Design temporal stability metrics (smoothness, break detection, persistence)
- Create simulation framework with multiple replications
- Prepare visualization functions for coefficient paths and performance

Exploring

Grepped dynamic_scad in .

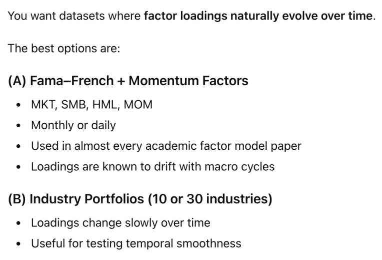
It seems like the dynamic_scad function is documented but not implemented in the current codebase. The user might be asking me to design an experiment for a dynamic SCAD method that they plan to implement, or perhaps it's in a different branch/version

Figure 8: Enter Caption

10 Real Data for both static and dynamic Scad

How AI Helps in This Setting

AI played an important role in designing and executing the real-data analysis for both static and Dynamic SCAD. It assisted me in identifying financial datasets where factor loadings are known to evolve over time—such as Fama–French factors, industry portfolios, and ETF return panels—and explained why these datasets are appropriate for evaluating temporal regularization. AI also guided the construction of rolling windows, scaling procedures, and alignment of predictors with returns to ensure that the dynamic model receives clean and properly structured inputs.



You want datasets where **factor loadings naturally evolve over time**.

The best options are:

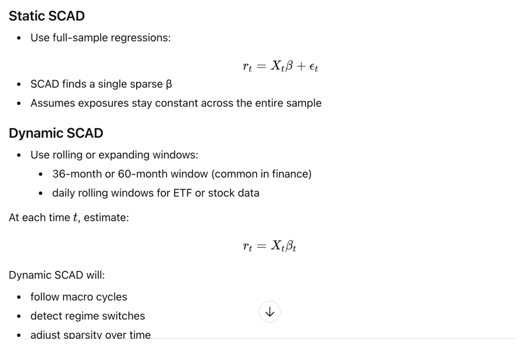
(A) Fama–French + Momentum Factors

- MKT, SMB, HML, MOM
- Monthly or daily
- Used in almost every academic factor model paper
- Loadings are known to drift with macro cycles

(B) Industry Portfolios (10 or 30 industries)

- Loadings change slowly over time
- Useful for testing temporal smoothness

Figure 9: Enter Caption



Static SCAD

- Use full-sample regressions:

$$r_t = X_t\beta + \epsilon_t$$

- SCAD finds a single sparse β
- Assumes exposures stay constant across the entire sample

Dynamic SCAD

- Use rolling or expanding windows:
  - 36-month or 60-month window (common in finance)
  - daily rolling windows for ETF or stock data

At each time t, estimate:

$$r_t = X_t\beta_t$$

Dynamic SCAD will:

- follow macro cycles
- detect regime switches
- adjust sparsity over time

Figure 10: Enter Caption