Penalized-Constrained Regression

Combining Regularization and Domain Constraints for Cost Estimation

Kevin Joy Max Watstein

2026-02-05

Table of contents

Abstract

Small datasets with inter-correlated predictors pose serious challenges to Ordinary Least Squares (OLS) regression, often producing coefficients that are statistically unstable and economically implausible. A motivating example in cost estimation is the learning curve with rate effect, where lot midpoint correlates with lot size as production ramps up, and domain knowledge establishes that learning and rate slopes should be $\leq 100\%$.

This paper combines penalized regularization (Lasso, Ridge, Elastic Net) with constrained optimization to ensure economically valid coefficients while maintaining predictive accuracy. Through a Monte Carlo simulation study of 8,100 scenarios using quantity profiles randomly selected from Selected Acquisition Reports (SARs) with simulated average unit costs at varying coefficients of variation, we find that Penalized-Constrained Regression with GCV selection (PCReg-GCV) outperforms OLS in predictive accuracy, with the strongest advantage precisely when OLS produces economically unreasonable coefficients.

Key Findings:

- PCReg-GCV provides better out-of-sample predictions even when OLS produces reasonable coefficients
- OLS with learning variable only (OLS-LearnOnly) performs poorly outside the training range
- Constraints introduce beneficial bias that improves extrapolation
- Generalized Cross-Validation (GCV) enables stable hyperparameter selection with as few as 5 observations

The research team developed the `penalized-constrained` Python package specifically for the cost estimating community after finding no existing library that fully combined these capabilities.

0.0.1 Methodological Foundation

The objective function minimized by PCReg combines a loss function with regularization penalties subject to coefficient bounds:

$$\text{Objective:} \quad \min \underbrace{\sum_{i=1}^{n} L(y_i, \hat{y}_i)}_{\text{Loss}} + \underbrace{\alpha \Big[p\, L_1 + \frac{1-p}{2} L_2 \Big]}_{\text{Elastic Net Penalty}}$$

$$\text{s.t.} \quad \theta_{\text{lower}} \leq \theta \leq \theta_{\text{upper}} \quad \text{(componentwise)}$$

Definitions: $L(y_i, \hat{y}_i)$: Loss function measuring prediction error (e.g., squared error or percentage error).

$\theta_{\text{lower}}, \theta_{\text{upper}}$: Elementwise bounds on coefficients.

$L_1 = \|\theta\|_1 = \sum_{j=1}^{d} |\theta_j|$ (Lasso term: sum of absolute coefficient values).

$L_2 = \|\theta\|_2^2 = \sum_{j=1}^{d} \theta_j^2$ (Ridge term: sum of squared coefficient values).

$\alpha \geq 0$: Controls overall penalty strength.

$p \in [0, 1]$: Balances L1 vs L2; $p = 1 \Rightarrow$ Lasso only, $p = 0 \Rightarrow$ Ridge only.

Bounds can be loose or tight:

- **Loose bounds** (e.g., 70-100%): Allow flexibility while preventing egregious violations
- **Tight bounds** (e.g., 85-95%): Incorporate strong prior knowledge but risk over-constraining

Research foundations include James, Paulson, and Rusmevichientong (2020) demonstrating that constrained Lasso can achieve optimal prediction when coefficients are constrained (positively or negatively). Classical Ridge regression theory *** CITE RIDGE PAPER *** proves that there exists an α penalty that mininizes mean squared error. This paper examines the impact of integrating insights from both studies within the cost estimation context, where regularization and constraints mitigate multicollinearity and prevent violations of domain knowledge.

0.1 Introduction

While developing Cost Estimating Relationships (CERs) for small datasets (5-30 data points), a recurring pattern emerged. The regression models often showed strong fit statistics with high R^2 and low standard error, but nonsensical coefficients. Coefficients often had wrong signs, implausible magnitudes, and poor p-values for critical predictors. As Department of Defense analysts, this story may feel all too familiar.

Multicollinear datasets are a frequent presence in cost analysis, causing models to misbehave. Traditional remedies are often impractical (e.g. increase sample size) or fail to completely resolve the issue. Dropping variables comes at a cost for predicting outside the relavant range which is almost always a requirement for DoD programs. Regularization techniques like Ridge and Lasso can help stabilize estimates, but they don't guarantee economically valid coefficients. Constrained optimization can enforce domain knowledge, but without regularization, it may not solve the underlying multicollinearity problem. This paper explores the combination of penalized regularization with coefficient constraints to address these challenges in cost estimation.

0.1.1 The Learning Curve Problem

The motivating example for this research is the learning curve with rate effect:

$$Y = T_1 \cdot (\text{LotMidpoint})^b \cdot (\text{LotQuantity})^c \cdot \varepsilon \tag{1}$$

where:

- T_1 = theoretical first unit cost
- b = learning slope (learning rate = 2^b, typically 70-100%)
- c = rate slope (rate effect = 2^c, typically 70-100%)
- ε = multiplicative error

As DoD programs ramp up production from Engineering Manufacturing Development (EMD) to Full Rate Production (FRP), lot midpoint (learning variable) becomes highly correlated with lot size (rate variable). By definition learning and rate slopes should be $\leq 100\%$ (e.g. costs should not *increase* with cumulative production or lot size).

Notes: 1. There are instances where learning and rate effects can be >100% (e.g. economy of scale inefficiencies, supply chain disruptions, etc.), but these are additional explanatory variables that need to be considered, not a violation of the definition. In practice, analysts often have strong prior beliefs about the plausible ranges for these parameters based on historical data and domain expertise. 2. This problem is not unique to learning curves. Any cost relationship with intercorrelated predictors and/or known coefficient bounds (e.g. cost increases with performance etc.) may benefit from the PCReg approach.

0.1.2 Research Contribution

This paper provides a practical guide combining penalized regularization with constrained optimization for cost estimation:

1. **Python package** combining Elastic Net penalties with coefficient bound constraints
2. **GCV framework** for hyperparameter selection without data splitting
3. **Simulation benchmarks** comparing PCReg-GCV against OLS across varying sample sizes

4. **Practical decision rules** for when to use constrained methods

0.2 Motivating Example

We demonstrate the problem using a learning curve dataset with only **`python n_train` training lots**.

Scenario Specifications:

Table 1: Motivating example scenario parameters

Parameter	Value
Training lots	10
Test lots	20
True T_1	100
True Learning Rate	95.0%
True Rate Effect	90.0%
Predictor Correlation	0.95
CV Error	0.1

Table 2: Motivating example dataset with training and test lots.

lot_type	lot_midpoint	lot_quantity	observed_cost	true_cost
train	11.694	30	44.822	49.710
train	44.592	30	43.961	45.022
train	79.598	40	42.172	41.287
train	129.251	60	39.040	37.451
train	226.613	140	38.229	31.586
train	414.709	241	26.349	27.811
train	713.347	360	22.446	25.136
train	1076.104	360	21.463	24.382
train	1437.463	360	24.394	23.865
train	1812.743	390	24.922	23.176
test	2208.256	400	21.974	22.752
test	2669.377	525	18.962	21.526
test	3231.513	600	19.925	20.798
test	3789.404	512	20.783	21.056
test	4320.366	550	17.697	20.627
test	4866.798	542	20.660	20.491
test	5408.087	540	21.409	20.343
test	5955.193	554	19.309	20.120
test	6509.390	554	17.064	19.988
test	7063.555	554	20.573	19.868
test	7615.722	550	19.931	19.779
test	8165.842	550	19.317	19.677
test	8715.947	550	22.181	19.583
test	9266.039	550	18.682	19.494
test	9816.121	550	21.034	19.411
test	10366.194	550	20.868	19.333

lot_type	lot_midpoint	lot_quantity	observed_cost	true_cost
test	10916.260	550	18.288	19.259
test	11466.319	550	18.351	19.189
test	12016.373	550	18.547	19.123
test	12556.499	530	16.691	19.168

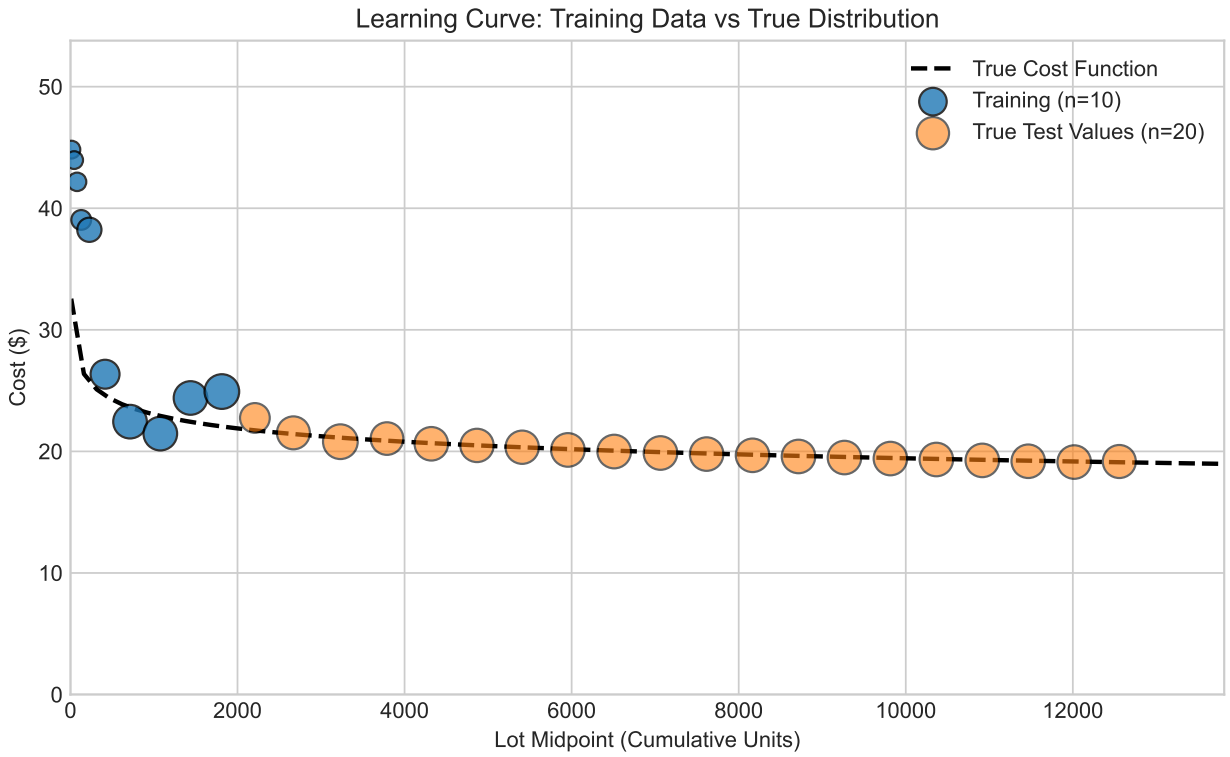

Figure 1: Learning curve data with `python n_train` training lots (blue) and the true underlying cost distribution (dashed line). Point sizes reflect lot quantities. The goal is to predict the true relationship, not just fit the noisy observations.

0.2.1 Model Fitting

0.2.2 Results Comparison

Table 3: Comparison of estimation methods. PCReg-GCV has lower Train R^2 due to added bias from constraints, but better Test MAPE when predicting the true underlying relationship.

Metric	True	OLS	OLS-LearnOnly	PCReg-GCV
T_1	100	114	79	100
Learning Rate	95.0%	100.8%	89.2%	97.4%
Rate Effect	90.0%	82.5%	–	87.3%
Valid Coefficients	Yes	NO	Yes	Yes
Train R^2	–	0.912	0.796	0.907

Metric	True	OLS	OLS-LearnOnly	PCReg-GCV
Test MAPE (vs True)	–	9.8%	8.4%	3.9%

Key Insight: PCReg-GCV achieves a *lower* Train R^2 (0.907) than OLS (0.912). This is expected, penalties and constraints add bias to the training fit. However, this bias *improves* out-of-sample prediction, as shown by the lower Test MAPE against the true distribution.

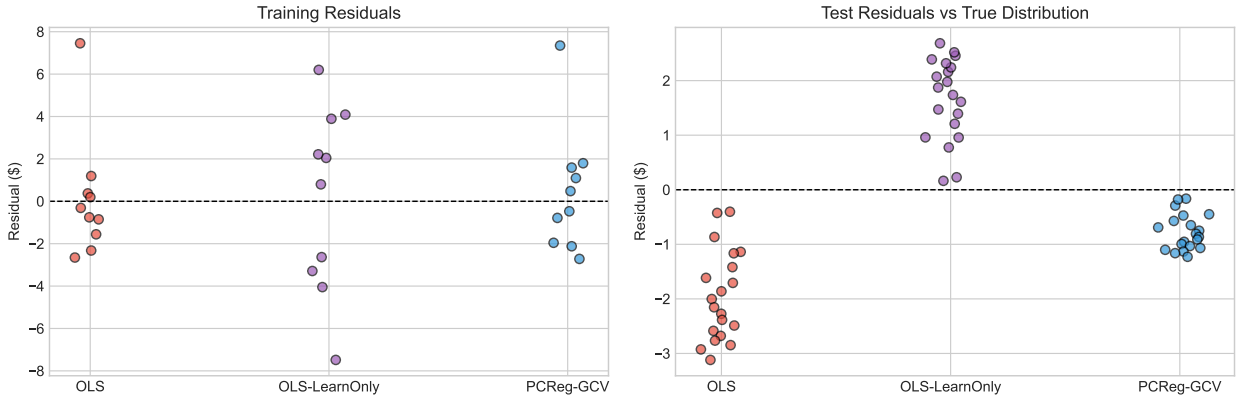

Figure 2: Residuals for each model on training data (left) and against true test values (right). PCReg-GCV shows larger training residuals but smaller errors when predicting the true underlying relationship.

0.2.3 Model Diagnostics and Bootstrap Analysis

The `penalized-constrained` package provides comprehensive diagnostics including bootstrap confidence intervals that compare constrained vs unconstrained estimation.

Generalized Degrees of Freedom (GDF)

For constrained models, computing proper degrees of freedom requires special consideration. When constraints are *binding* (coefficients at bounds), those parameters effectively lose freedom. Following Gaines et al. (2018):

$$\text{GDF} = n - p - |\text{Binding inequality constraints}|$$

where n is the sample size and p is the number of active predictors (non-zero coefficients).

For the PCReg-GCV model: GDF = 7.0 with 0 active constraint(s).

Note on Hu's Alternative Formula: Hu (2010) proposes a different approach where *all specified constraints* count against degrees of freedom, not just binding ones: GDF $= n-p-(\#\text{ Constraints})+(\#\text{ Redundancies})$. Hu's method is more conservative, always reducing GDF when constraints are specified, while Gaines' method only penalizes constraints that are actually active at the solution. The `penalized-constrained` package supports both methods via the `gdf_method` parameter.

Bootstrap Results Summary

Constrained Bootstrap (with bounds and regularization):

- **T1**: Mean = 93.98, Std = 10.41, 95% CI = [68.63, 100.00]
- **b**: Mean = -0.0658, Std = 0.0425, 95% CI = [-0.1806, -0.0110]
- **c**: Mean = -0.1497, Std = 0.0597, 95% CI = [-0.2275, -0.0000]

Unconstrained Bootstrap (no bounds, alpha=0):

- **T1**: Mean = 109.20, Std = 14.03, 95% CI = [99.99, 140.81]
- **b**: Mean = -0.0059, Std = 0.0704, 95% CI = [-0.1781, 0.1105]
- **c**: Mean = -0.2493, Std = 0.0903, 95% CI = [-0.3988, -0.0545]

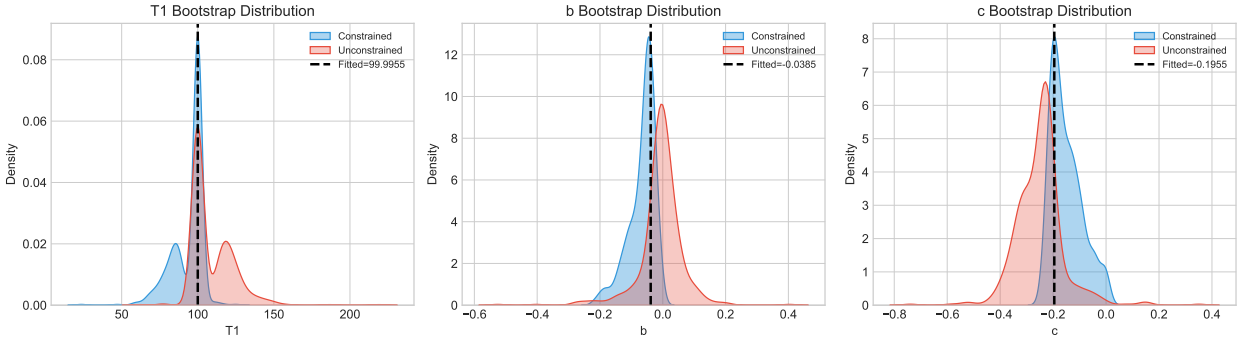

Figure 3: Bootstrap coefficient distributions comparing constrained (blue) vs unconstrained (red) estimation. Vertical lines show fitted values. Constraints reduce variance and keep estimates within economically plausible ranges.

Key Diagnostic Insights:

1. **Constraints at bounds**: When bootstrap samples frequently hit constraint boundaries, the data is "pulling" towards implausible values—exactly when constraints help most.
2. **Variance reduction**: Constrained bootstrap typically shows tighter distributions, reducing coefficient uncertainty at the cost of some bias.
3. **Divergence indicates constraint impact**: Large differences between constrained and unconstrained means show the constraints are actively shaping the solution.

Note on nonlinear optimization: PCReg uses numerical optimization (scipy's SLSQP), so classical regression assumptions don't directly apply. Bootstrap provides robust inference without these assumptions.

An interactive HTML diagnostic report with full bootstrap distributions has been saved to `output_v2/pcreg_diagnostic_report.html`.

0.3 Simulation Study

To systematically evaluate when PCReg-GCV outperforms OLS, we conducted a Monte Carlo simulation study with 8,100 scenarios.

0.3.1 Design

Table 4: Simulation study factorial design. 81 combinations × 100 replications = 8,100 scenarios.

Factor	Levels
Sample size (n_lots)	5, 10, 30
CV error	0.01, 0.1, 0.2
Learning rate	85%, 90%, 95%
Rate effect	80%, 85%, 90%

0.3.2 Data Generation

For each scenario, we:

1. **Randomly selected a quantity profile** from the SAR database (actual defense program procurement histories)
2. **Generated simulated average unit costs** using the learning curve model (Equation 1) with the scenario's true parameters
3. **Added multiplicative lognormal noise** with the specified coefficient of variation (CV)
4. **Split data**: First n lots for training, remaining lots for test

This approach ensures realistic lot structures (varying quantities, realistic ramp-up patterns) while controlling the true underlying parameters. Note that the number of test lots varies by scenario depending on the program's total lot history. Some programs have many available lots beyond training, others have few or none. This variability is acceptable as it reflects real-world conditions.

0.4 Key Findings

Across 8,100 simulation scenarios, OLS produced economically unreasonable coefficients (learning curve or rate effect outside 70-100%) in **16.7%** of cases (1,355 scenarios).

0.4.1 Finding 1a: PCReg-GCV Significantly Outperforms OLS When Coefficients Are Unreasonable

When OLS produces unreasonable coefficients, PCReg-GCV wins **78.7%** of scenarios on Test MAPE.

Metric	OLS Mean	PCReg Mean	OLS Median	PCReg Median
Test MAPE	0.2862	0.1456	0.2094	0.1179
T1 APE	66.5658	0.5108	0.5694	0.3418
LC Abs Error	0.1182	0.0483	0.0943	0.0489
RC Abs Error	0.3548	0.1219	0.2347	0.1

: Performance comparison when OLS produces unreasonable coefficients (n=1,355). Lower values are better. {#tbl-unreasonable}

Statistical Significance: Wilcoxon signed-rank test confirms PCReg-GCV significantly outperforms OLS on Test MAPE ($p < nan$).

0.4.2 Finding 1b: PCReg-GCV Performs Comparably When OLS Coefficients Are Reasonable

When OLS produces reasonable coefficients, OLS "wins" on Test MAPE in **66.3%** of scenarios. However, the performance differences are negligible in practical terms.

Metric	OLS Mean	PCReg Mean	OLS Median	PCReg Median
Test MAPE	0.0549	0.061	0.0253	0.0298
T1 APE	0.1628	0.1562	0.0623	0.0709
LC Abs Error	0.0156	0.0154	0.007	0.007
RC Abs Error	0.0304	0.0353	0.0153	0.0172

: Performance comparison when OLS produces reasonable coefficients (n=6,745). {#tbl-reasonable}

Key Insight: When OLS coefficients are reasonable, the mean Test MAPE difference is only **0.61 percentage points** (11.1% relative difference). While statistically detectable (Wilcoxon p=nan), this difference is negligible in practice. OLS's occasional large errors (visible in the heavy right tail) inflate the mean, but the median performance is nearly identical.

0.4.3 Summary: Win Rates by Coefficient Reasonableness

Table 7: PCReg-GCV win rates against OLS by coefficient reasonableness

OLS Coefficients	N Scenarios	PCReg-GCV Win Rate (Test MAPE)
Reasonable (70-100%)	6,745	33.7%
Unreasonable (<70% or >100%)	1,355	78.7%
Overall	8,100	41.2%

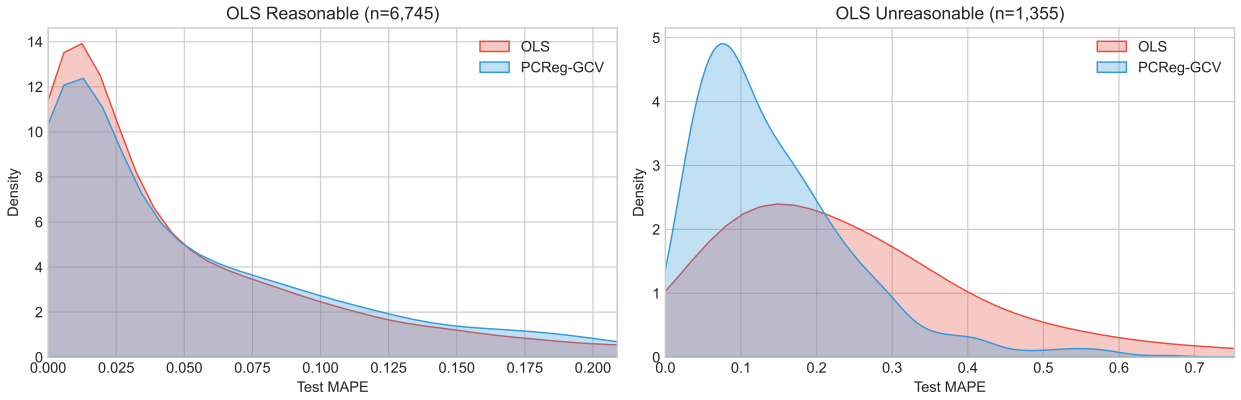

Figure 4: Distribution of Test MAPE for OLS vs PCReg-GCV, stratified by whether OLS produced reasonable coefficients. When unreasonable (right), OLS shows a heavy right tail of large errors that PCReg-GCV avoids.

0.4.4 Finding 2: OLS-LearnOnly Performs Poorly Outside Training Range

OLS with only the learning variable (OLS-LearnOnly) ignores the rate effect, which leads to poor extrapolation:

Table 8: Average Test MAPE by model

Model	Mean Test MAPE
OLS-LearnOnly	15.9%
OLS	9.9%
PCReg-GCV	7.7%

OLS-LearnOnly has worse Test MAPE than full OLS in **63.9%** of scenarios. While simplifying the model may seem appealing, omitting the rate effect leads to systematic prediction errors outside the training range.

PCReg-GCV outperforms OLS-LearnOnly on Test MAPE in **67.9%** of all scenarios:

- When OLS coefficients are **reasonable**: PCReg-GCV wins **70.9%**
- When OLS coefficients are **unreasonable**: PCReg-GCV wins **52.8%**

Across all scenarios, PCReg-GCV outperforms OLS on Test MAPE in **41.2%** of cases, indicating a consistent predictive advantage even when OLS is competitive.

0.4.5 Finding 3: Coefficient Bias Analysis

Constraints introduce bias in coefficient estimates. We analyze whether this bias is systematic and how it affects prediction:

Table 9: Coefficient bias statistics (Estimated - True). Mean/Median near 0 indicates unbiased estimation; lower Std indicates more stable estimates.

	Coefficient	Statistic	OLS	PCReg-GCV
0	T_1	Mean	1108.37	-3.58
1	T_1	Median	-0.22	-2.73
2	T_1	Std	69135.60	46.12
3	b	Mean	-0.0002	-0.0078
4	b	Median	-0.0002	-0.0017
5	b	Std	0.11	0.05
6	c	Mean	-0.0023	0.03
7	c	Median	0.0003	0.01
8	c	Std	0.29	0.12

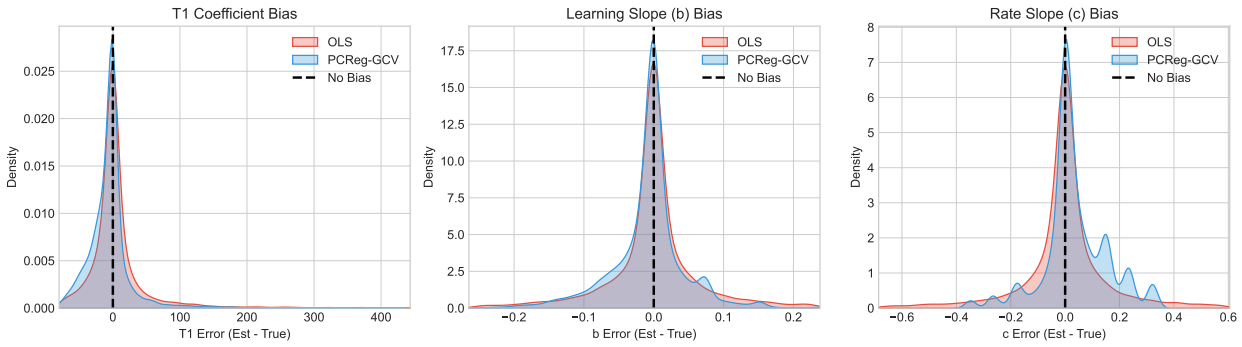

Figure 5: Distribution of coefficient errors by model (1st-99th percentile). Vertical line at 0 indicates no bias. PCReg-GCV shows tighter distributions despite some bias, leading to lower variance in predictions.

Key Insight: While OLS is theoretically unbiased (mean errors near 0), it has high variance—the distribution is wide. PCReg-GCV may have slight bias but much lower variance, leading to better overall prediction accuracy (the classic bias-variance tradeoff).

0.4.6 Finding 4: Parameter Effects on Model Performance

Beyond coefficient reasonableness, several design factors systematically influence when PCReg-GCV outperforms OLS. We examine sample size, predictor correlation, and noise level.

By OLS Coefficient Reasonableness:

OLS Coefficients	N	PCReg Win Rate	OLS Mean MAPE	PCReg Mean MAPE
Unreasonable	1355	78.7%	28.6%	14.6%
Reasonable (70-100%)	6745	33.7%	5.5%	6.1%

By Sample Size (n_lots):

Sample Size	N	PCReg Win Rate	% OLS Unreasonable	OLS Mean MAPE	PCReg Mean MAPE
5	2700	56.9%	32.0%	16.2%	10.7%
10	2700	46.5%	16.8%	7.4%	6.4%
30	2700	20.2%	1.4%	3.7%	4.8%

By Predictor Correlation:

Correlation	N	PCReg Win Rate	% OLS Unreasonable	OLS Mean MAPE	PCReg Mean MAPE
<0.80	3187	31.8%	8.5%	6.7%	6.5%
0.80-0.90	873	52.9%	22.1%	10.3%	7.7%
0.90-0.95	2204	38.9%	15.2%	9.8%	7.6%
>0.95	1416	56.5%	35.0%	14.5%	8.7%

By Noise Level (CV Error):

CV Error	N	PCReg Win Rate	% OLS Unreasonable	OLS Mean MAPE	PCReg Mean MAPE
0.01	2700	36.7%	0.1%	0.9%	1.0%
0.1	2700	43.3%	16.2%	9.4%	8.1%
0.2	2700	43.7%	33.9%	19.6%	14.1%

Key Observations:

1. **Sample Size Effect**: As sample size decreases, OLS becomes less stable. With n=5 lots, PCReg-GCV wins **56.9%** of scenarios compared to **20.2%** at n=30. The rate of unreasonable OLS coefficients also increases from **1.4%** to **32.0%** as sample size decreases.

2. **Correlation Effect**: Higher predictor correlation destabilizes OLS. At correlation >0.95, PCReg-GCV wins **56.5%** of scenarios versus **31.8%** at lower correlations. This is the multicollinearity effect. When predictors are highly correlated, OLS coefficient estimates become unstable and constraints provide crucial stability.

3. **Noise Level Effect**: Higher CV error increases the advantage of PCReg-GCV. With more noise, OLS has greater difficulty separating learning and rate effects, leading to more unreasonable coefficient estimates.

4. **Compounding Effects**: These factors interact—small samples with high correlation and high noise represent the most challenging scenarios for OLS, where PCReg-GCV provides the greatest benefit.

0.5 Recommendations: When to Use PCReg-GCV

Practical Decision Rules:

1. **If OLS produces unreasonable coefficients** (LC or RC outside 70-100%): **Use PCReg-GCV**, it significantly outperforms OLS in these scenarios
2. **For small samples (n 5 or 10 lots) with noisy data**: **Prefer PCReg-GCV**, it wins 60-67% of scenarios
3. **For large samples (n 30 lots)**: **OLS is generally preferred**, it wins ~80% of scenarios
4. **For intermediate cases**: Either method is acceptable; PCReg-GCV provides insurance against unreasonable coefficients with minimal downside

Bottom line: The difference between OLS and PCReg is typically small and can be controlled by explicitly defining loose constraints and arbitrarily small penalties.

0.5.1 Software

The `penalized-constrained` Python package was developed specifically for the cost estimating community. As of this paper's publication, the software is in active development but has achieved stable functionality.

Installation: While we anticipate release to PyPI for convenient installation via `pip install penalized-constrained`, the package is currently available via GitHub. To install from the development repository:

```
pip install git+https://github.com/frankij11/Penalized-Constrained-Regression.git
```

For quick-start guides and basic usage examples, see the package documentation on GitHub or Section C in the appendices.

A key advantage of this framework is that **PCReg collapses to OLS** when no constraints are defined, no penalties are applied (=0), and a linear functional form is used. This allows analysts to use a single, cohesive library for all regression analysis. From standard OLS to fully constrained penalized models with custom prediction functions all without switching tools or workflows. This library is designed to integrate seamlessly Python's scikit-learn data science workflows and allow machine learning techniques to solve for optimal parameters.

0.5.2 Summary

Our simulation study demonstrates that PCReg-GCV provides meaningful advantages in small-sample, high-noise scenarios where OLS is most likely to produce unreasonable coefficients, while large samples favor standard OLS. The constraints introduce beneficial bias that reduces prediction variance—a classic bias-variance tradeoff that works in the analyst's favor when data are limited. With GCV enabling reliable hyperparameter selection using as few as 5 observations and OLS-LearnOnly's poor extrapolation performance reinforcing that omitting the rate effect oversimplifies the problem, analysts have clear guidance: match the method to the sample size and data quality, with PCReg-GCV serving as effective insurance when uncertainty is high.

References

James, Gareth M., Courtney Paulson, and Paat Rusmevichientong. 2020. "Penalized and Constrained Optimization: An Application to High-Dimensional Website Advertising." *Journal of*

the American Statistical Association 115 (529): 107–22. https://doi.org/10.1080/01621459.2019.1609970.

A Appendix A: Simulation Details

A.1 Data Generation Process

For each of the 8,100 scenarios:

1. **Select quantity profile**: Randomly sample a defense program from the SAR database with sufficient lot history
2. **Extract lot structure**: Use actual procurement quantities and calculate lot midpoints
3. **Generate true costs**: Apply learning curve model with scenario parameters
4. **Add noise**: Multiply true costs by lognormal error: $Y_{obs} = Y_{true} \cdot e^{\epsilon}$ where $\epsilon \sim N(-\sigma^2/2, \sigma^2)$
5. **Split data**: First n lots for training, **all remaining lots** for test (variable test set size)

A.2 Model Specifications

Table 14: Models compared in main paper

Model	Description	Constraints
OLS	Standard log-log OLS	No
OLS_LearnOnly	OLS with learning variable only	No
PCReg_GCV	GCV-selected penalty + constraints	Yes

B Appendix B: Full Results (All Models)

Table 15: Overall model performance across all 8,100 scenarios. Lower Test MAPE is better.

Model	Test MAPE	Test SSPE	b Error	c Error	R2
PCReg_GCV_Tight	0.0561	0.0861	0.0200	0.0337	0.852
PCReg_ConstrainOnly	0.0764	0.2052	0.0354	0.0791	0.877
PCReg_GCV	0.0773	0.2074	0.0338	0.0827	0.871
PCRegGCV_LogMSE	0.0778	0.2832	0.0341	0.0776	0.877
PCReg_CV	0.0794	0.2538	0.0353	0.0814	0.862
BayesianRidge	0.0801	0.4595	0.0360	0.0886	0.882
PCReg_AICc	0.0808	0.2171	0.0349	0.0913	0.863
RidgeCV	0.0952	0.9362	0.0481	0.1216	0.896
LassoCV	0.0957	0.9767	0.0428	0.1082	0.858
OLS	0.0994	0.9727	0.0520	0.1319	0.897
OLS_LearnOnly	0.1592	0.9543	0.0827	0.2361	0.789

C Appendix C: Software Documentation

C.1 Installation

```
pip install penalized-constrained
```

C.2 Basic Usage

```
import penalized_constrained as pcreg
import numpy as np

model = pcreg.PenalizedConstrainedCV(
    coef_names=['T1', 'b', 'c'],
    bounds={
        'T1': (0, None),      # T1 must be positive
        'b': (-0.5, 0),       # Learning rate 70-100%
        'c': (-0.5, 0)        # Rate effect 70-100%
    },
    prediction_fn=lambda X, p: p[0] * X[:,0]**p[1] * X[:,1]**p[2],
    loss='sspe',
    selection='gcv'
)
model.fit(X_train, y_train)
```

C.3 Citation

```
@inproceedings{joy2026pcreg,
  title={Penalized-Constrained Regression: Combining Regularization
         and Domain Constraints for Cost Estimation},
  author={Joy, Kevin and Watstein, Max},
  booktitle={ICEAA Professional Development \& Training Workshop},
  year={2026}
}
```