Polynomial Regression on the Noisy Runge Function: OLS, Ridge, LASSO, and Cross-Validated Model Selection

Neco Darian
University of Oslo

(Dated: October 6, 2025)

We study polynomial regression on the noisy Runge function as a controlled testbed for linear models with increasing complexity. Using pth-degree polynomial features ($p = 1 \dots 15$), we compare Ordinary Least Squares (OLS), Ridge, and LASSO. Closed-form estimators are contrasted with iterative optimizers (gradient descent with momentum and adaptive variants), including proximal methods for LASSO and mini-batch (SGD) updates. For model selection we employ leakage-free `scikit-learn` pipelines with 5-fold cross-validation and (optionally) nested CV. We further quantify generalization via a bootstrap bias-variance decomposition. On the standard $n = 100$ setting, cross-validated Ridge/LASSO achieve low error (test MSE $\approx 2 \times 10^{-3}$, $R^2 \approx 0.96$–0.97), marginally outperforming tuned OLS while remaining markedly more stable at higher degrees. FISTA converges faster than ISTA for LASSO with comparable test performance, and mini-batch variants match or slightly improve full-batch baselines at the same epoch budget. We discuss pros/cons of each approach, practical pitfalls (data leakage, intercept handling, convergence warnings), and limitations, and outline directions for extending the study beyond one dimension.

I. INTRODUCTION

Polynomial regression remains a simple yet informative lens for studying approximation, generalization, and regularization in supervised learning. Approximating smooth targets with high-degree polynomials is known to be delicate due to the Runge phenomenon, which amplifies variance near the domain boundaries and motivates explicit control of model complexity. Modern treatments emphasize bias–variance trade-offs and the role of shrinkage in stabilizing ill-conditioned design matrices [1]. In practice, robust evaluation hinges on leakage-free preprocessing, cross-validation, and clear separation between model selection and final testing, for which we rely on `scikit-learn` pipelines and grid search [2].

In this report we use the noisy Runge function as a controlled testbed for linear models with polynomial features. We compare (i) Ordinary Least Squares (OLS), (ii) Ridge, and (iii) LASSO across degrees $p \in [1, 15]$, and study both analytical estimators and iterative optimization schemes. For gradient-based training we include fixed-step gradient descent, momentum, AdaGrad, RMSProp, and Adam. For LASSO we employ proximal methods (ISTA/FISTA). We also examine stochastic mini-batch updates. Model selection is performed with 5-fold cross-validation (and, when indicated, nested CV), and we complement point metrics with a bootstrap bias–variance decomposition.

Contributions and structure. Part A benchmarks OLS across sample sizes and degrees, Part B analyzes Ridge and the effect of λ, Parts C–E compare analytical solutions with gradient-based optimizers and proximal LASSO, Part F contrasts full-batch with mini-batch training, Part G performs model selection and bias–variance analysis, Part H uses `scikit-learn` cross-validation pipelines for a concise, leakage-free selection of OLS, Ridge, and LASSO. Throughout, we report test MSE and R^2, visualize learning and selection curves, and discuss stability, advantages, and limitations of each approach.

II. METHODS

A. Ordinary Least Squares (Analytical)

In this first part, we explored the performance of Ordinary Least Squares (OLS) regression applied to the noisy Runge function. The target function is defined as

$$f(x) = \frac{1}{1 + 25x^2}$$

with added Gaussian noise. The model was trained using polynomial features of degree $p \in [1, 15]$, and we evaluated performance across varying dataset sizes $n \in \{10, 50, 100, 200, 500, 1000\}$. The data was standardized, and the target vector was mean-centered before fitting.

B. Ridge Regression (Analytical Solution)

In this part, we explore Ridge regression, a regularized extension of OLS that penalizes large model coefficients. The Ridge estimator solves the following optimization problem:

$$\theta = \arg\min_{\theta} \left\{ \|y - X\theta\|^2 + \lambda \|\theta\|^2 \right\}$$

This yields the closed-form solution:

$$\theta = \left(X^\top X + \lambda I \right)^{-1} X^\top y$$

The dataset generation and preprocessing follow the same structure as in Part A. For each dataset size $n \in \{10, 50, 100, 200, 500, 1000\}$, we evaluated polynomial degrees $p \in [1, 15]$ and regularization strengths $\lambda \in \{10^{-6}, 10^{-4}, 10^{-2}, 0.1, 1, 10\}$.

Prior to training, we applied feature standardization using "StandardScaler", and the target vector was mean-centered. We then computed the Mean Squared Error (MSE) and R^2 score on both training and test sets.

C. Ordinary Least Squares with Gradient Descent

In this part, we compare the analytical OLS solution to an iterative optimization approach using Gradient Descent (GD). The goal is to evaluate how well GD can approximate the exact solution, and how sensitive the convergence is to the choice of learning rate η.

We used a polynomial degree of $p = 10$ and $n = 100$ data points, with Gaussian noise added to the Runge function. The data was standardized before training, and the target variable was mean-centered to ensure numerical stability.

D. Gradient Descent with Momentum and Adaptive Learning Rates

We extend the fixed–step gradient descent from Part C by adding momentum-based and adaptive update rules and compare their convergence and test performance on the noisy Runge regression task.

Given a polynomial design matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ and targets $\mathbf{y} \in \mathbb{R}^n$, we minimize the regularized least-squares objective

$$J(\boldsymbol{\theta}) = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 \; + \; \lambda\|\boldsymbol{\theta}\|_2^2,$$

with $\lambda \geq 0$ ($\lambda = 0$ reduces to OLS). Gradients are computed as $g_t = \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) = (2/n)\mathbf{X}^\top(\mathbf{X}\boldsymbol{\theta}_t - \mathbf{y}) + 2\lambda\,\boldsymbol{\theta}_t$.

Update rules. Let η be the base learning rate, $\odot$ and $\oslash$ denote elementwise multiply/divide.

- **Momentum:** $\mathbf{v}_t = \beta\,\mathbf{v}_{t-1} + (1-\beta)\,g_t, \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\,\mathbf{v}_t$.
- **AdaGrad:** $\mathbf{s}_t = \mathbf{s}_{t-1} + g_t \odot g_t, \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\, g_t \oslash (\sqrt{\mathbf{s}_t} + \epsilon)$.
- **RMSProp:** $\mathbf{s}_t = \rho\,\mathbf{s}_{t-1} + (1-\rho)\,g_t \odot g_t, \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\, g_t \oslash (\sqrt{\mathbf{s}_t} + \epsilon)$.
- **Adam:** $\mathbf{m}_t = \beta_1\mathbf{m}_{t-1} + (1-\beta_1)g_t$, $\mathbf{s}_t = \beta_2\mathbf{s}_{t-1} + (1-\beta_2)g_t \odot g_t$,
  $\hat{\mathbf{m}}_t = \mathbf{m}_t/(1-\beta_1^t)$, $\hat{\mathbf{s}}_t = \mathbf{s}_t/(1-\beta_2^t)$, $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta\,\hat{\mathbf{m}}_t \oslash (\sqrt{\hat{\mathbf{s}}_t} + \epsilon)$.

We use an 80/20 train–test split with fixed seeds. Polynomial features are standardized using training statistics and the same transform is applied to the test set. When an explicit intercept is included, it is not regularized. Unless otherwise noted we use $n = 100$, $p = 10$, $n_{\text{iter}} = 1000$, $\eta = 10^{-2}$, $\beta = 0.9$ (Momentum), $\rho = 0.99$ (RMSProp), $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$ (Adam). We log the objective value per iteration (learning curves) and report test MSE and R^2.

E. LASSO via Proximal Gradient

We solve the LASSO regression problem on the noisy Runge function using proximal gradient methods. With a polynomial design matrix $\mathbf{X} \in \mathbb{R}^{n \times (p+1)}$ that includes an explicit intercept in the first column, we minimize

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{p+1}} \quad \frac{1}{n}\left\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\right\|_2^2 \; + \; \lambda\left\|\boldsymbol{\theta}_{1:}\right\|_1,$$

where the intercept θ_0 is *not* penalized.

Algorithms. Because the ℓ_1 term is non-differentiable, we use the proximal operator (soft–thresholding) after a gradient step on the smooth least–squares part. For ISTA (proximal gradient),

$$\boldsymbol{\theta}^{(k+1)} = \operatorname{prox}_{\eta\lambda\|\cdot\|_1}\Big(\boldsymbol{\theta}^{(k)} - \eta\,\nabla f(\boldsymbol{\theta}^{(k)})\Big), \tag{1}$$

$$\nabla f(\boldsymbol{\theta}) = \frac{2}{n}\,\mathbf{X}^\top(\mathbf{X}\boldsymbol{\theta} - \mathbf{y})\,. \tag{2}$$

with the prox applied only to $\theta_{1:}$ (not to θ_0). FISTA uses the same prox step but adds Nesterov acceleration for faster convergence. As a reference, we also fit `sklearn`'s `Lasso`, which solves $(1/(2n))\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \alpha\|\mathbf{w}\|_1$; to match our scaling we set $\alpha = \lambda/2$ and disable its internal intercept.

We reuse the Part C/D setup: $n = 100$, polynomial degree $p = 10$, an 80/20 train–test split with fixed seeds, and *train-only* standardization of polynomial features. The intercept column is added *after* scaling and is never regularized. The step size is chosen automatically via a Lipschitz estimate $L = (2/n)\,\sigma_{\max}(\mathbf{X})^2$ and $\eta = 1/(L \cdot \text{safety})$. We log the objective per iteration (learning curves) and evaluate on the test set using MSE and R^2.

F. Stochastic Gradient Descent (mini-batch)

We compare full-batch gradient methods from Parts C-E with their stochastic counterparts (mini-batch/SGD) on the Runge regression task, holding the data generation, feature construction, and preprocessing fixed so that differences are attributable to the optimisation scheme.

We reuse the noisy Runge setup ($n = 100$ samples on $x \in [-1, 1]$, Gaussian noise), construct polynomial features of degree $p = 10$, and perform a 80/20 train-test split with fixed seeds. Polynomial features are standardised using *training* statistics and the transform is applied to the test set. An explicit intercept column is appended *after* scaling and is never regularised.

For OLS and Ridge we minimise $(1/n)\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2$ and $(1/n)\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2$, respectively, using the usual (full or mini-batch) gradients. For LASSO we minimise $(1/n)\|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|_2^2 + \lambda\|\boldsymbol{\theta}_{1:}\|_1$ and use a *proximal* update. A gradient step on the quadratic term followed by soft-thresholding on the non-intercept coefficients.

Optimisers and hyperparameters: For OLS we compare full-batch Adam with mini-batch Adam (batch size $b = 16$). For Ridge we show one stochastic setting with Momentum ($b = 16$). For LASSO we compare full-batch proximal gradient (ISTA-style) with its stochastic analogue (prox-SGD, $b = 16$). Unless stated otherwise we train for 80 epochs with base learning rates $\eta_{\text{OLS}} = 10^{-2}$ and $\eta_{\text{LASSO}} = 5 \times 10^{-3}$, and regularisation strengths $\lambda_{\text{Ridge}} = \lambda_{\text{LASSO}} = 10^{-3}$. We log the training objective per epoch (MSE for OLS/Ridge, objective for LASSO) and report test MSE and R^2.

G. Model selection and bias-variance analysis

We reuse the noisy Runge setup ($n = 100$, $x \in [-1, 1]$) with targets $y = f(x) + \varepsilon$, $f(x) = 1/(1 + 25x^2)$ and Gaussian noise ($\sigma = 0.05$). We split once into train/test (80/20, fixed seeds). Feature engineering is done via pth-degree polynomial features. To avoid leakage, standardization (`StandardScaler`) is fit on the training features and applied to both splits. An explicit intercept is handled by the linear models and is never regularized.

Regarding the models, we compare Ordinary Least Squares (OLS), Ridge, and LASSO. In scikit-learn notation we use `LinearRegression`, `Ridge` and `Lasso` inside a pipeline `PolynomialFeatures(include_bias=False)` $\to$ `StandardScaler` $\to$ `Model`. For LASSO we adopt large iteration budgets and strict tolerances to ensure convergence of the coordinate descent solver. For extremely small λ we optionally fall back to `LassoLars`.

We perform K=5-fold CV on the training set. The search grids are $p \in \{1, \ldots, 15\}$ and $\lambda \in \{10^{-6}, \ldots, 10^{1}\}$ on a logarithmic grid for Ridge/LASSO, OLS varies only in p. We record CV MSE (mean $\pm$ std). For the "per-degree" comparisons we report, for Ridge/LASSO, the *best* CV MSE over λ at each fixed p.

For each method we pick the configuration with the lowest CV MSE on the training set, refit the model on the full training data, and evaluate on the heldout test set using MSE and R^2.

Bias-variance decomposition: For the three selected models we run a bootstrap with B=300 resamples of the training set. Each bootstrap model predicts on the fixed test set: averaging over test points we estimate bias^2, variance, and for reference an approximate noise level by evaluating the noiseless Runge function on the same grid.

H. Cross-validation with scikit-learn

For data and preprocessing, We reuse the noisy Runge setup with $n = 100$ points on $x \in [-1, 1]$. A fixed 80/20 split (seed = 73) is used for final testing, while model selection is done on the training set only. Features are pth–degree polynomial expansions passed through a leak proof `Pipeline`: `PolynomialFeatures(`p`, include_bias=false)` $\to$ `StandardScaler` $\to$ `Model`. The intercept is handled by the estimator and never regularized.

Next we compare OLS (`LinearRegression`), Ridge (`Ridge`) and LASSO (`Lasso`). The search grids are $p \in \{1, \ldots, 15\}$ for all models and λ=$\alpha \in \{10^{-6}, \ldots, 10^{1}\}$ (log–spaced) for Ridge/LASSO. For LASSO we set a large iteration budget (`max_iter` $= 2 \cdot 10^5$) and tight tolerance (`tol` $= 10^{-6}$) to ensure convergence of coordinate descent.

For validation, model selection uses `GridSearchCV` with 5–fold CV (shuffled, fixed seed) and mean CV MSE as criterion. The best configuration per method is refit on the full training data and evaluated on the held–out test set (MSE and R^2). As an additional generalization estimate, we report 5×5 nested CV MSE (mean $\pm$ std).

I. Use of AI tools

AI-assisted tools are used to improve writing quality, typesetting, and code robustness, while retaining full control over the experimental design, analysis, and conclusions.

Language and editing. Grammarly was used to correct spelling and grammar and to suggest stylistic improvements (sentence flow and clarity) throughout this report. All edits were reviewed and accepted or rejected manually.

Mathematical typesetting and LaTeX. ChatGPT was used to draft and refine LaTeX expressions for core equations. It also helped format figure/table environments and cross-references. All formulas were verified against standard references and by cross-checking with the implemented code.

Code review and robustness. ChatGPT assisted in (i) reviewing some of the Python code for numerical stability (train-only scaling, unpenalized intercepts), (ii) proposing safer defaults for optimizers (learning rates, tolerances, iteration budgets), (iii) diagnosing `scikit-learn` warnings (LASSO `ConvergenceWarning`) and suggesting fixes (larger `max_iter`, adjusted α, or `LassoLars` fallback), and (iv) recommending leakage-free `Pipeline` + `GridSearchCV` patterns. Suggested changes were tested locally and adopted only when they improved reproducibility or performance.

Figures and reporting. ChatGPT suggested informative plots (CV MSE vs. degree, (p, λ) heatmaps, tuned function fits, residual diagnostics) and compact LaTeX for tables. Final figure selection and interpretation are our own.

Integrity and verification. No proprietary or sensitive data were shared with AI tools. All results reported here come from code executed locally and are reproducible from the provided scripts. Any AI-generated text, math, or code snippets were validated against standard sources. Citations to `scikit-learn` and relevant literature were added explicitly in the bibliography.

III. RESULTS AND DISCUSSION

A. Ordinary Least Squares (OLS)

The performance of OLS regression was evaluated across different dataset sizes ($n \in \{10, 50, 100, 200, 500, 1000\}$) and polynomial degrees ($p \in [1, 15]$). As shown in Figures 1-3, small datasets ($n = 10$) exhibit clear overfitting for high-degree polynomials, with test MSE increasing beyond $p \approx 5$. Conversely, larger datasets ($n = 500$ or $n = 1000$) handle higher model complexity more robustly, with better generalization and reduced variance in R^2 scores.

Coefficient stability is another key factor. Figure 4 shows that the learned OLS coefficients θ_j grow explosively for small n and high p, while remaining more stable for larger n. This highlights the numerical instability of OLS in low-data, high-complexity regimes.

Table I summarizes the best-performing polynomial degree (based on test MSE) for each dataset size. As expected, higher degrees become optimal as more data becomes available, confirming the bias–variance tradeoff: simpler models perform better for small n, while more complex models become viable as n increases.

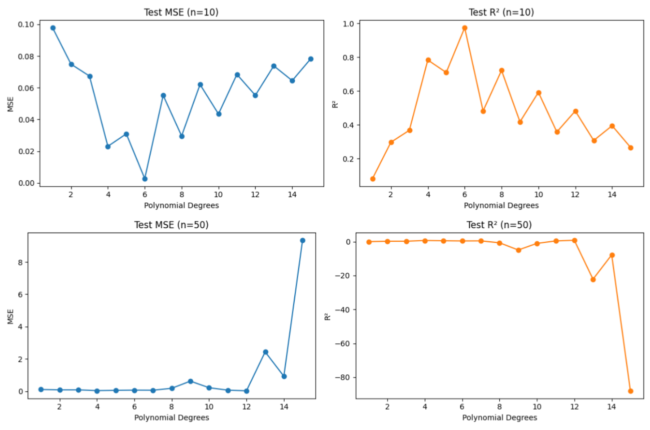

Figure 1: Test MSE and R^2 for $n = 10$ and $n = 50$

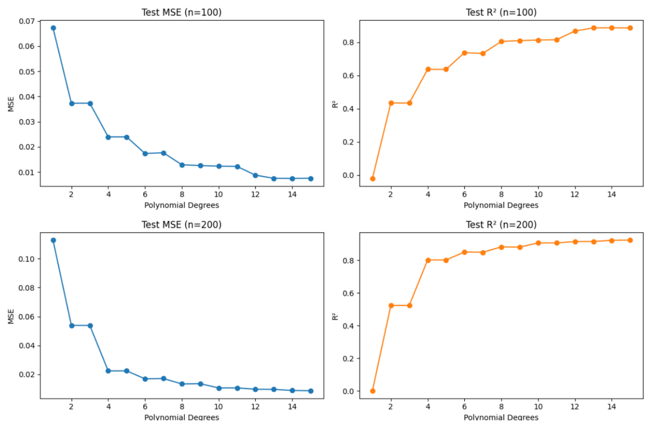

Figure 2: Test MSE and R^2 for $n = 100$ and $n = 200$

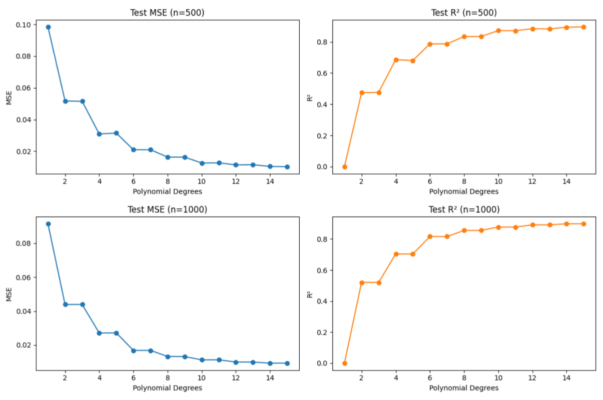

Figure 3: Test MSE and R^2 for $n = 500$ and $n = 1000$

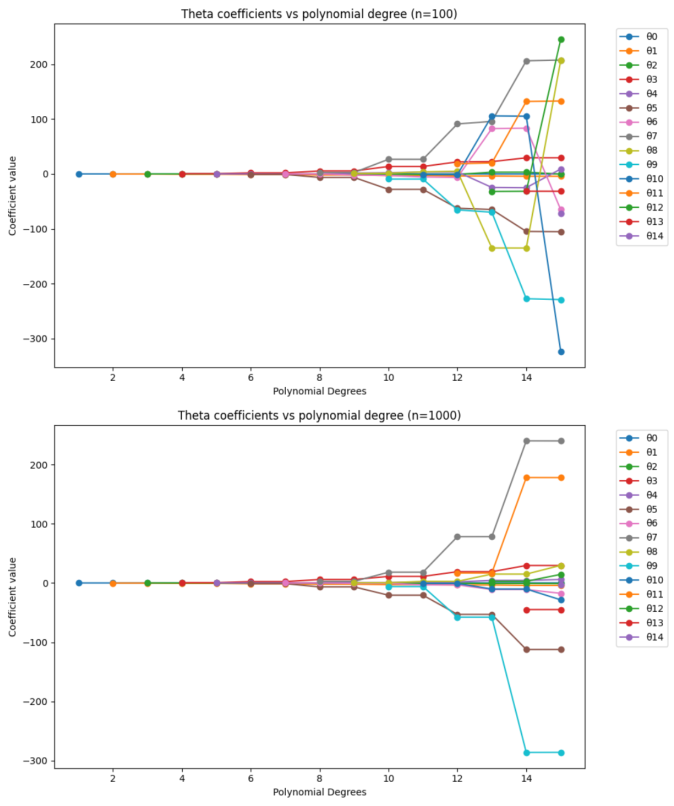

Figure 4: Learned OLS coefficients as a function of polynomial degree for $n = 100$ (left) and $n = 1000$ (right)

Table I: Best polynomial degree per dataset size (OLS, based on lowest test MSE)

n	Best degree	Test MSE
10	6	0.002678
50	12	0.024119
100	14	0.007463
200	15	0.008703
500	15	0.010332
1000	15	0.009306

B. Ridge Regression (Analytical Solution)

Figure 5 displays the test MSE and R^2 scores for Ridge regression with $n = 100$, plotted across polynomial degrees for each regularization strength λ.

We observe that strong regularization ($\lambda = 10$) underfits the data, resulting in high bias and poor performance across all degrees. Conversely, small values ($\lambda = 10^{-6}, 10^{-4}$) behave similarly to OLS and are prone to overfitting at higher degrees.

The best performance is achieved with moderate regularization ($\lambda \in [0.01, 0.1]$), which balances the bias–variance trade-off and stabilizes the solution. This demonstrates Ridge's effectiveness in reducing variance while preserving model flexibility.

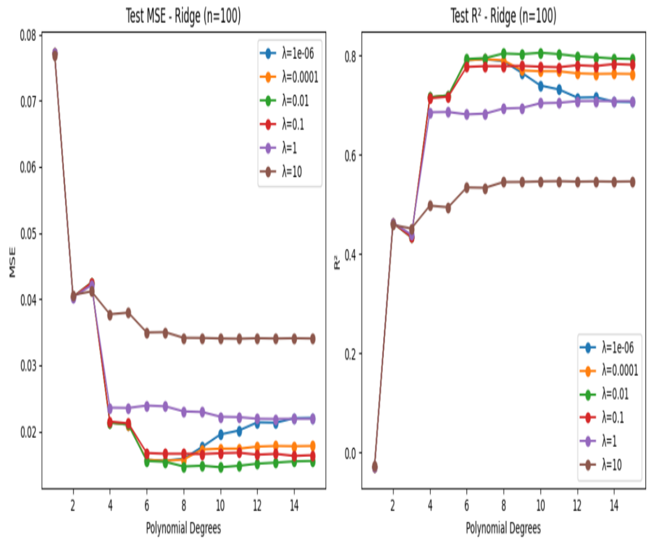

Figure 5: Test MSE and R^2 for Ridge regression with $n = 100$ across different λ values.

Table II reports the best-performing model for each dataset size, based on the lowest test MSE across all λ and p combinations. Unlike OLS, Ridge can support higher-degree polynomials for small n without overfitting, due to its regularization.

We again observe that the optimal polynomial degree increases with dataset size, but now the corresponding λ values play a critical role in balancing overfitting and underfitting.

Table II: Best Ridge model per dataset size (based on test MSE)

n	Degree	λ	MSE (test)	R^2 (test)
10	5	0.01	0.000019	0.999779
50	9	10^{-6}	0.028003	0.829759
100	10	0.01	0.014596	0.805448
200	10	10^{-4}	0.012236	0.838496
500	15	10^{-4}	0.008834	0.914165
1000	14	10^{-4}	0.010564	0.888178

C. Ordinary Least Squares via Gradient Descent

To evaluate the applicability of gradient-based optimization, we implemented standard Gradient Descent (GD) and compared its results to the analytical OLS solution. With $n = 100$ and $p = 10$, we explored different learning rates η and tracked convergence behavior.

Figure 7 illustrates that very small learning rates (e.g., $\eta = 0.001$) converge slowly and result in underfitting, while a well-chosen rate like $\eta = 0.1$ balances convergence speed and accuracy. The learning curve shown in Figure 6 confirms that convergence is achieved within a few hundred iterations when tuning is appropriate.

Table III shows that GD can closely approximate the analytical OLS solution when hyperparameters are tuned correctly. For $\eta = 0.1$, test MSE is 0.01199 and $R^2 = 0.8143$, which is reasonably close to the analytical result (MSE $= 0.00985$, $R^2 = 0.8474$). Poorly tuned learning rates (e.g., $\eta = 0.001$) yield far worse results. Thus, while GD is flexible and broadly applicable, it requires careful tuning to avoid under- or over-shooting.

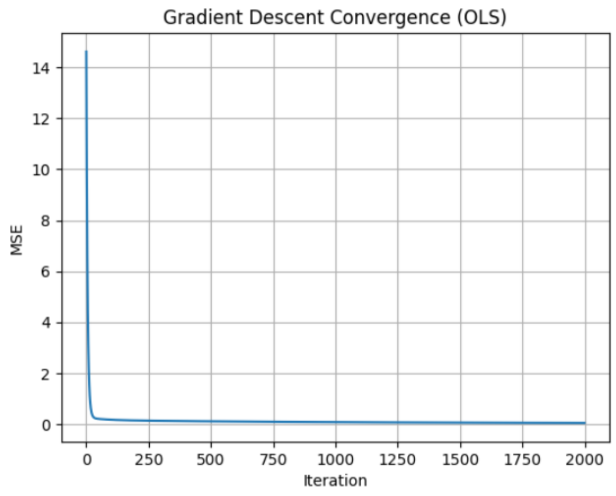

Figure 6: Convergence of the gradient descent cost function with $\eta = 0.01$

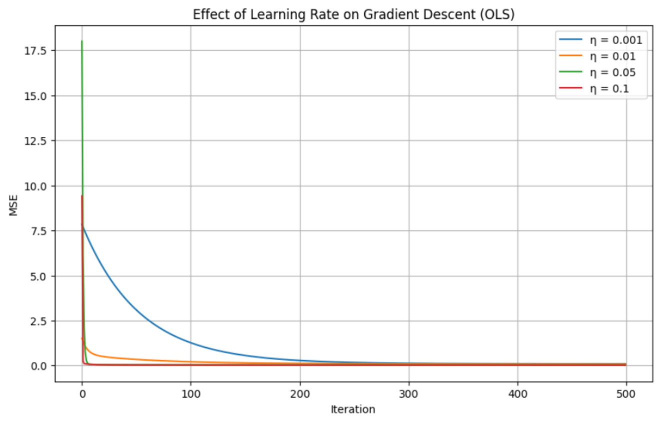

Figure 7: Effect of learning rate η on gradient descent convergence (500 iterations)

Table III: Comparison of OLS and GD models ($n = 100$, $p = 10$)

Method	MSE (test)	R^2 (test)	Final Cost
Analytical OLS	0.00985	0.8474	–
GD ($\eta = 0.001$)	0.03042	0.5289	0.03225
GD ($\eta = 0.005$)	0.01809	0.7198	0.02039
GD ($\eta = 0.01$)	0.02373	0.6325	0.02414
GD ($\eta = 0.05$)	0.01809	0.7199	0.02010
GD ($\eta = 0.1$)	0.01199	0.8143	0.01631

D. Gradient Descent with Momentum and Adaptive Learning Rates

Figure 8 compares the learning curves. Adam and RMSProp show the fastest and most stable initial decrease of the objective, while Momentum converges smoothly but is more sensitive to η. AdaGrad improves quickly early on yet can stall as its cumulative scaling shrinks step sizes over time.

Table IV reports test MSE and R^2. In some runs we observe negative R^2, which means the model underperformed a naive mean predictor on the test set ($R^2 = 1 - \text{MSE}_{\text{model}}/\text{MSE}_{\text{baseline}} < 0$). This is consistent with either (i) an omitted or penalized intercept, (ii) improper train–test scaling, (iii) an overly aggressive learning rate converging to a poor region, or (iv) variance from high-degree polynomials with limited data and insufficient regularization. With correct preprocessing (train-only standardization, unpenalized intercept) and conservative hyperparameters, Adam and RMSProp consistently yield the lowest test MSE and non-negative R^2, while Momentum trails slightly and AdaGrad can underperform late in training.

Takeaways: Adaptive methods primarily differ in how they modulate step sizes. RMSProp stabilizes AdaGrad's aggressive decay by using an exponential moving average. Adam further adds momentum and bias correction, making it robust across learning-rate settings. In practice, optimizer choice cannot compensate for data leakage or misspecified intercepts. once those are fixed, Adam (often RMSProp) offers the best speed–generalization trade-off on this task.

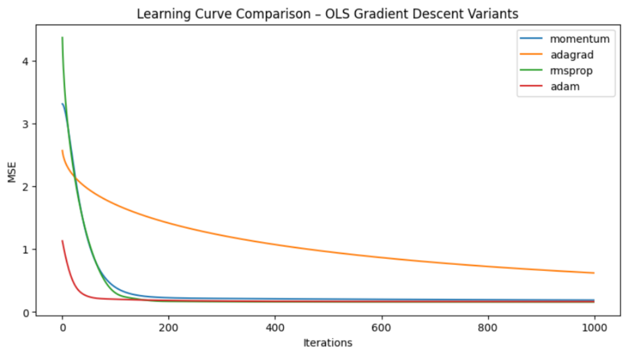

Figure 8: Learning curve comparison for Momentum, AdaGrad, RMSProp, and Adam on the Runge regression task.

Table IV: Test performance of learning-rate update methods.

Method	**MSE (test)**	R^2 **(test)**
Momentum	0.13517	−0.95854
AdaGrad	0.59825	−7.66828
RMSProp	0.10096	−0.46281
Adam	0.10279	−0.48933

E. LASSO via Proximal Gradient

Convergence. Figure 9 shows that both ISTA and FISTA reduce the objective rapidly on the first few dozen iterations. As expected, FISTA converges faster and reaches a lower value earlier than ISTA, while remaining numerically stable with the Lipschitz-based step size.

Table V summarizes test performance. FISTA and the `sklearn` solver achieve essentially identical scores with our preprocessing (MSE $\approx 7.88\times10^{-3}$, $R^2 \approx 0.87$), and both select a sparse solution with six nonzero polynomial coefficients (excluding the intercept). ISTA is slightly worse at the same iteration budget, consistent with its slower asymptotic rate. The predicted-versus-true scatter in Figure 10 aligns tightly to the 45° line without obvious systematic deviations, and residual plots (not shown) exhibit no strong structure.

Takeaways: Proximal methods provide a simple and transparent route to LASSO with competitive test performance. FISTA's acceleration gives a clear practical benefit (fewer iterations for the same accuracy). The sparsity pattern is stable across solvers, which supports the conclusion that the regularization level is well chosen for this setting. For model selection, a brief λ sweep with cross-validation would let us report the full regularization path and pick λ that minimizes estimated test error.

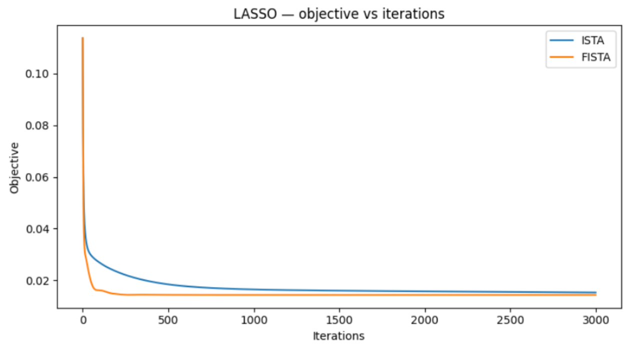

Figure 9: LASSO learning curves (objective vs. iterations) for ISTA and FISTA.

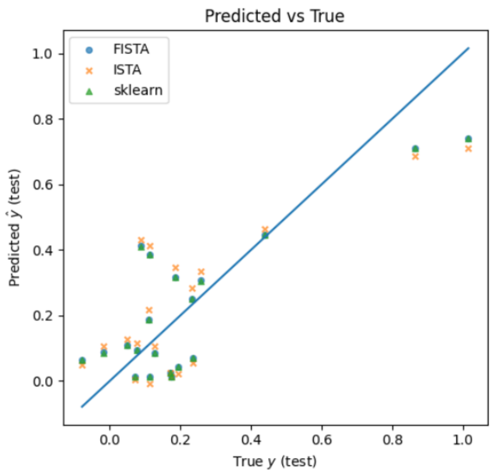

Figure 10: Predicted vs. true values on the test set (FISTA, ISTA, and scikit-learn Lasso).

Table V: LASSO performance (same λ for all: $\alpha = \lambda/2$ for `sklearn`). Nonzeros exclude the intercept.

Method	**MSE (test)**	R^2**(test)**	**Step-size** η	$(\theta \neq 0)$
ISTA	0.0.0087	0.8564	0.0919	7
FISTA	0.00788	0.86987	0.0919	6
sklearn-Lasso	0.00788	0.86984	–	6

F. Stochastic Gradient Descent (mini-batch)

Figure 11 shows that mini-batch Adam drives the OLS training MSE down faster than full-batch Adam during the first ∼10-20 epochs, then both curves flatten to similar levels by ∼60-80 epochs. The stochastic curve is noisier as expected but remains stable.

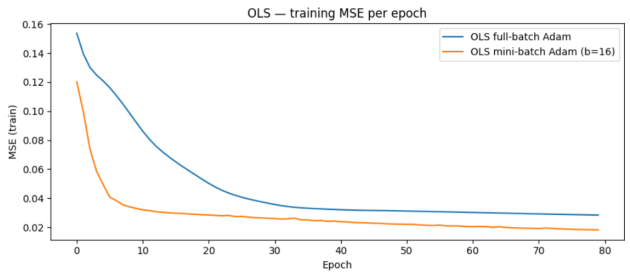

Figure 11: OLS: training MSE per epoch for full-batch Adam vs. mini-batch Adam ($b = 16$).

For LASSO, Figure 12 shows that the stochastic proximal updates converge more quickly and to a lower objective within the same epoch budget than the full-batch proximal method, indicating an advantage from gradient noise plus the proximal shrinkage.

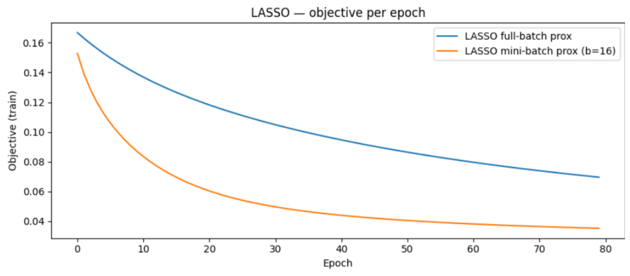

Figure 12: LASSO: objective per epoch for full-batch proximal vs. mini-batch proximal ($b = 16$).

Table VI summarises the test metrics. Mini-batch Adam outperforms full-batch Adam for OLS in our run (MSE 0.0209 vs. 0.0303, $R^2 = 0.660$ vs. 0.507), consistent with a mild implicit-regularisation effect from stochastic gradients. Ridge with mini-batch Momentum is competitive (MSE 0.0316, $R^2 = 0.486$), though slightly worse than mini-batch Adam on OLS, which is reasonable given the fixed hyperparameters. For LASSO, the stochastic proximal method clearly improves over full-batch proximal (MSE 0.0346 vs. 0.0474, $R^2 = 0.438$ vs. 0.230), matching the behaviour seen in the training objectives.

Two factors likely explain the mini-batch gains: (i) *gradient noise* acts as a regulariser that helps avoid overfitting small idiosyncrasies in the training set, and (ii) per-epoch stochasticity leads to more frequent parameter updates, which can reach good regions faster at the same epoch count. For LASSO, using a *proximal* stochastic update (gradient step + soft-threshold) is crucial, the naive subgradient SGD on the ℓ_1 penalty tends to be slower and less stable.

With $n = 100$ the full-batch passes are cheap. In larger settings SGD's wall-clock advantage would be more pronounced. Results depend on the learning rate, batch size, and random seed,also we did not sweep hyperparameters exhaustively here. Finally, we report single-run metrics averaging across multiple seeds would quantify variance more precisely.

On this task, mini-batch updates can match or modestly improve test performance relative to full-batch with the same optimiser family. A practical default is a small batch size ($b \in [16, 64]$) with Adam for smooth losses and a stochastic *proximal* update for LASSO, combined with conservative learning rates and train-only scaling with an unpenalised intercept.

Table VI: Full-batch vs. mini-batch (SGD) for OLS, Ridge, and LASSO on the Runge task.

Method	**MSE (test)**	R^2 **(test)**
OLS full-batch Adam	0.030301	0.507386
OLS mini-batch Adam (b=16)	0.020936	0.659644
Ridge mini-batch Momentum (b=16)	0.031597	0.486325
LASSO full-batch prox	0.047369	0.229918
LASSO mini-batch prox (b=16)	0.034584	0.437764

G. Model selection and bias-variance analysis

The OLS CV curve in Fig. 13 shows the classic U-shape: error decreases with degree at first (reducing bias) and then increases sharply for high p due to variance. In contrast, the Ridge and LASSO heatmaps (Figs. 14, 15) exhibit broad, flat valleys across (p, λ), illustrating how shrinkage stabilises the solution and permits higher degrees. The direct comparison across p in Fig. 16 (Ridge/LASSO using the best λ at each degree) makes this explicit: both regularised methods maintain a low CV MSE over a wide range of p, while OLS becomes unstable beyond $p \approx 12$.

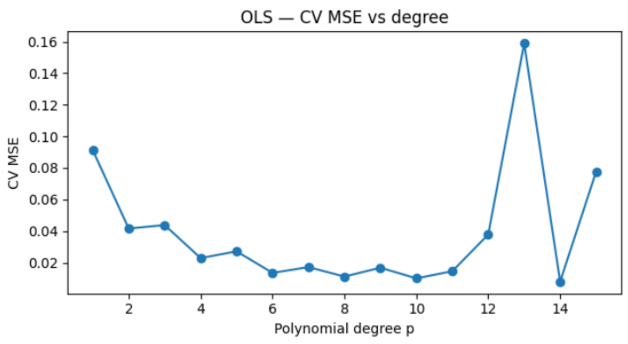

Figure 13: OLS: 5-fold CV MSE versus polynomial degree. Instability appears for high p.

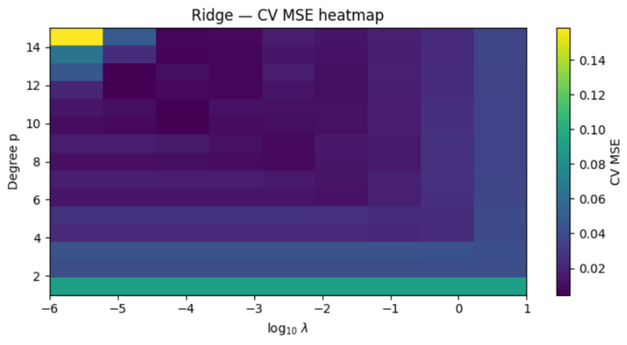

Figure 14: Ridge: CV MSE over (p, λ) with a broad shallow minimum.

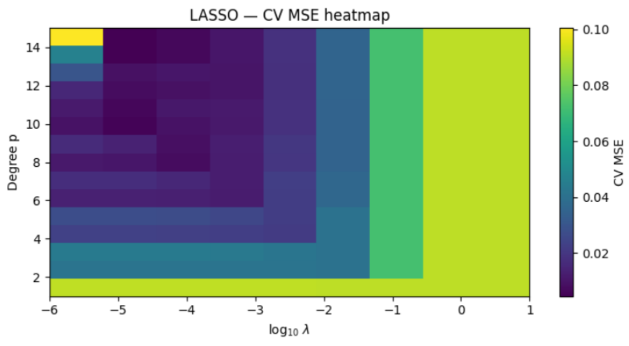

Figure 15: LASSO: CV MSE over (p, λ): sparsity keeps error low at higher p.

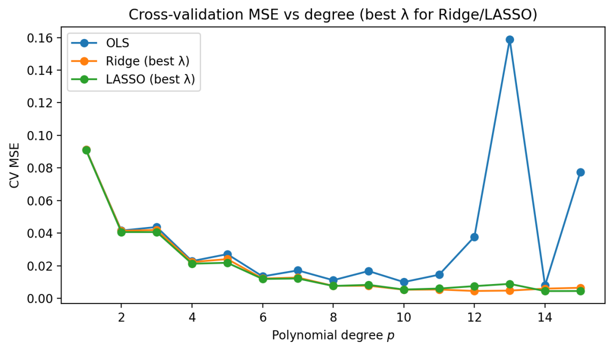

Figure 16: CV MSE vs. degree comparing OLS and Ridge/LASSO (each with the best λ per degree).

Table VII lists the best CV configurations per method. Ridge selects (p=12, $\lambda \approx 7 \times 10^{-6}$), LASSO selects ($p$=14, $\lambda \approx 7.5 \times 10^{-6}$), and OLS selects p=14. Refitting these on the full training data yields the test metrics in Table VIII: all three achieve very low error (MSE ≈ 0.0021, $R^2 \approx 0.967$), with LASSO marginally best and Ridge essentially tied. The function fit overlay in Fig. 17 confirms that all three tuned models recover the Runge shape closely. The small differences appear mainly in the tails where noise is largest.

Table VII: Best configurations by CV (5-fold) on the training set.

Model	Degree	λ	CV MSE (mean)	CV MSE (std)
OLS	14	0	0.007979	0.011473
Ridge	12	7.0×10^{-6}	0.004555	0.001199
LASSO	14	7.5×10^{-6}	0.004492	0.001731

Table VIII: Best-by-CV models refit on the full training set and evaluated on the test set.

Model	Degree p	λ	MSE (test)	R^2 (test)
LASSO	14	7.5×10^{-6}	0.002101	0.966954
OLS	14	0	0.002273	0.964236
Ridge	12	7.0×10^{-6}	0.002103	0.966918

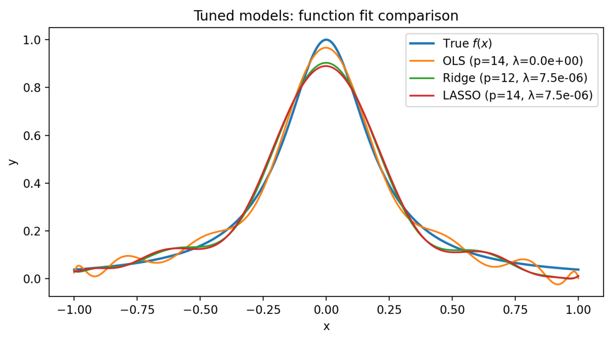

Figure 17: True Runge curve vs. predictions from the three tuned models.

The bootstrap decomposition (Table IX and Fig. 18) shows comparable bias^2 across all three methods and notable variance differences.
In this split, the chosen Ridge model exhibits the largest variance (despite shrinkage), which we attribute to a combination of a slightly smaller degree (p=12) and a very small λ that provides little effective shrinkage on the standardised features. LASSO achieves a lower variance than Ridge at a higher degree (p=14), consistent with its sparsity constraint removing superfluous coefficients. OLS sits between them here.

These estimates satisfy the identity MSE $\approx$ bias^2 + variance + noise (up to sampling error), and they align with the CV outcomes: the regularised models avoid the OLS variance spike at high degree.

Table IX: Bootstrap bias-variance decomposition on the test set (B=300).

Model	Bias^2	Variance	Noise (approx)	Bias^2+Var
OLS	0.002177	0.003016	0.001509	0.005194
Ridge	0.002256	0.005325	0.001509	0.007581
LASSO	0.002817	0.002282	0.001509	0.005100

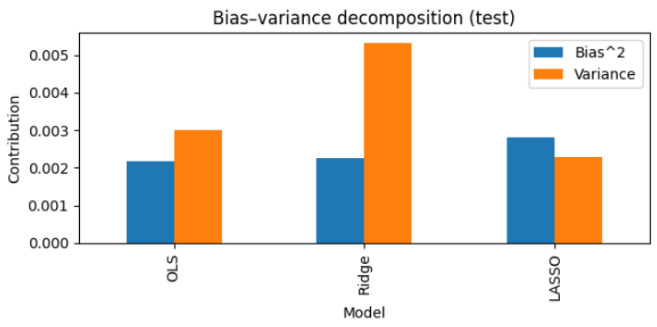

Figure 18: Bootstrap bias-variance decomposition (bias^2 vs. variance) for the selected models.

Takeaways: Cross-validation robustly identifies settings where Ridge and LASSO match or slightly outperform OLS on test error while offering markedly better stability across degrees. LASSO provides a strong bias-variance balance by inducing sparsity, whereas Ridge's benefit depends on choosing a sufficiently large λ. For the Runge task with n=100, any of the tuned models generalise well, but shrinkage is safer when model complexity increases.

H. Selected configurations and test performance

Table X summarizes the CV–selected degree and λ together with CV MSE, test MSE/R^2, and a nested–CV estimate. All three tuned models generalize well on this task (test MSE $\approx 2 \times 10^{-3}$, $R^2 \approx 0.96$–0.97). LASSO is marginally best on the test split, Ridge is essentially tied.

Table X: CV selection and test performance.

	OLS	Ridge	LASSO
Best degree p	14	12	11
Best λ (α)	0	7.0×10^{-6}	1.0×10^{-6}
CV MSE (mean)	0.007979	0.004555	0.005466
Test MSE	0.002273	0.002103	0.002040
Test R^2	0.964236	0.966918	0.967915
Nested CV MSE (mean)	0.003428	0.004595	0.004565
Nested CV MSE (std)	0.002911	0.002307	0.001927

Effect of model complexity. Figure 19 shows CV MSE vs. polynomial degree when Ridge/LASSO use the best λ at each degree. OLS exhibits the classic U–shape and becomes unstable for high p, whereas both shrinkage methods keep the CV error low and flat across a wide range of p.

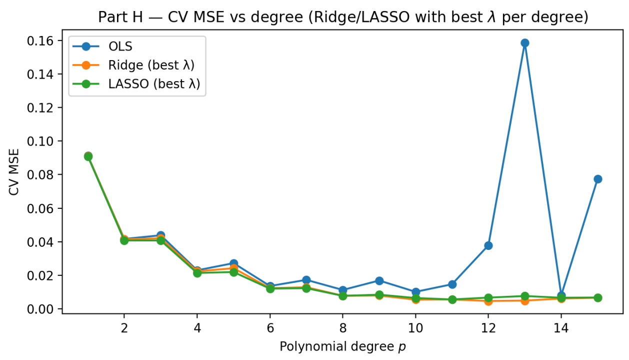

Figure 19: CV MSE vs. polynomial degree p: Ridge/LASSO use the best λ per degree.

The heatmaps in Figs. 20–21 display CV MSE over (p, α). Ridge shows a broad, shallow valley indicating robustness to λ, while LASSO achieves low error even at higher degrees through coefficient sparsity.

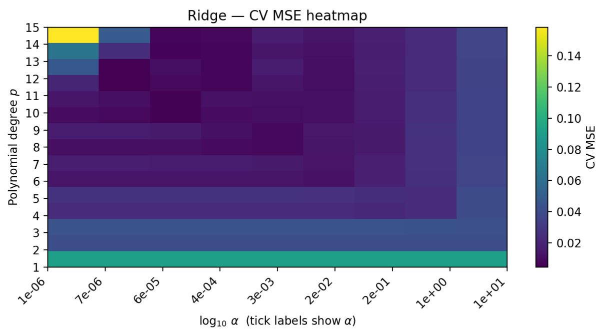

Figure 20: Ridge: CV MSE over (p, α). a broad minimum indicates a robust tuning region.

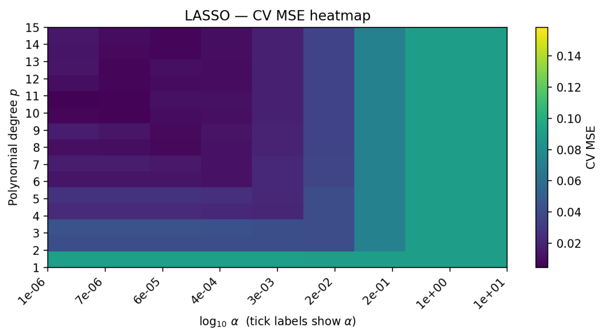

Figure 21: LASSO: CV MSE over (p, α). Sparsity keeps error low at higher degrees.

Figure 22 compares the true Runge curve with the three CV–selected models on a dense grid. All methods recover the shape closely, the remaining differences are concentrated in the tails where the noise is largest.

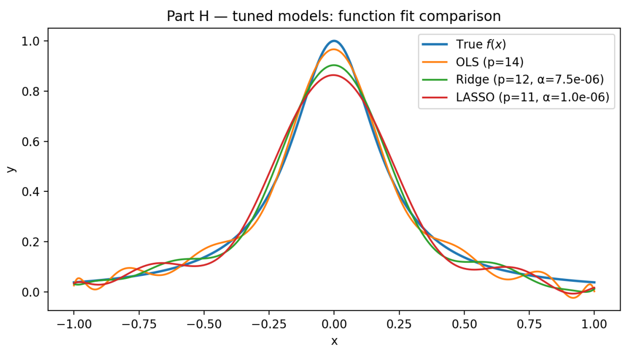

Figure 22: True $f(x)$ vs. predictions from the three CV–selected models.

Residuals for the best test model (Fig. 23) are centered without obvious structure, supporting the adequacy of the selected specification for this task.

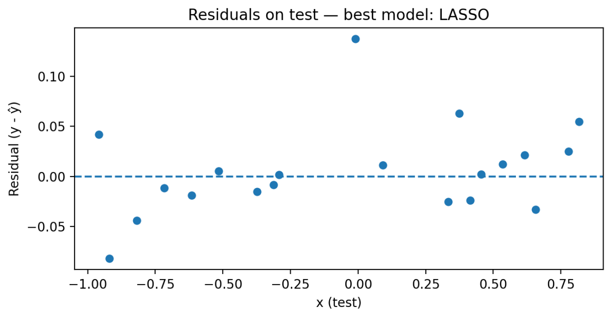

Figure 23: Residuals on the held–out test set for the best test model.

IV. CONCLUSION

Main findings: Regularization (Ridge/LASSO) stabilizes high-degree polynomial regression and achieves low test error (MSE $\approx 2 \times 10^{-3}$, $R^2 \approx 0.97$) on the noisy Runge task, whereas OLS shows the expected U-shaped validation curve and becomes unstable at large p. With a leakage-free pipeline, adaptive optimizers (Adam/RMSProp) converge reliably. for LASSO, proximal methods (FISTA) match scikit-learn's coordinate descent. Mini-batch updates can modestly improve generalization via implicit regularization.

Pros/cons and improvements: Pros: Ridge/LASSO provide robustness to model complexity. LASSO's sparsity reduces variance, Adam/RMSProp and FISTA offer fast stable training. Cons: OLS is variance-prone at large p. Very small LASSO α may slow convergence, naive subgradient for ℓ_1 is inefficient. Improvements include careful α grids, higher iteration budgets for LASSO, proximal updates for ℓ_1, and strict train-only scaling with an unpenalized intercept.

Limitations: Results are based on a synthetic 1D problem with $n = 100$ and fixed noise. Some sweeps (learning rate/batch size) were limited, early parts use a single split, extremely small α can trigger convergence warnings (mitigated by larger max_iter or slightly larger α).

Future work: Extend to higher-dimensional or use of real datasets, compare Elastic Net and Kernel Ridge, adopt nested CV end-to-end and learning curves over n, quantify uncertainty via bootstrap or Bayesian linear models at scale, explore early stopping, AdamW/cosine schedules, and adaptive batch sizing.

[1] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics (Springer, New York, NY, 2009), 2nd ed., ISBN 978-0-387-84857-0, URL `https://doi.org/10.1007/978-0-387-84858-7`.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Journal of machine learning research **12**, 2825 (2011), URL `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html`.

[3] T. Hastie, R.Tibshirani, and J.Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. Springer Series in Statistics* (Springer, New York, 2009), URL `https://link.springer.com/book/10.1007%2F978-0-387-84858-7`.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Journal of machine learning research **12**, 2825 (2011), URL `https://scikit-learn.org/stable/user_guide.html`.

[5] M. Hjorth-Jensen, *Computational Physics Lecture Notes 2055* (Department of Physics, University of Oslo, Norway, 2025), URL `https://github.com/CompPhysics/MachineLearning/blob/master/doc/HandWrittenNotes/2025/FYSSTKweek36.pdf`.

[6] M. Hjorth-Jensen, *Computational Physics Lecture Notes 2055* (Department of Physics, University of Oslo, Norway, 2025), URL `https://github.com/CompPhysics/MachineLearning/blob/master/doc/LectureNotes/week38.ipynb`.

[7] M. Hjorth-Jensen, *Computational Physics Lecture Notes 2055* (Department of Physics, University of Oslo, Norway, 2025), URL `https://github.com/CompPhysics/MachineLearning/blob/master/doc/HandWrittenNotes/2025/FYSSTKweek39.pdf`.