

Comparing regularisation paths of (conjugate) gradient estimators in ridge regression

Laura Hucker, Markus Reiß, Thomas Stark

Linear regression setting

Under random design

$$y_i = x_i^\top \beta_0 + \varepsilon_i, \quad i = 1, \dots, n$$

- $\beta_0 \in \mathbb{R}^p$ unknown true coefficient vector
- i.i.d. observations (x_i, y_i) of $\mathbb{R}^p$ -valued feature vectors x_i , $\mathbb{R}$ -valued responses y_i
- ε_i error variables, $\mathbb{E}[\varepsilon_i | x_i] = 0$, $\text{Var}(\varepsilon_i | x_i) = \sigma^2$, $\sigma > 0$
- $p \gg n$ possible (high-dim. setting)
- **short:** $y = X\beta_0 + \varepsilon$ with $y := (y_1, \dots, y_n)^\top$, $X := (x_1, \dots, x_n)^\top$,
 $\varepsilon := (\varepsilon_1, \dots, \varepsilon_n)^\top$
- assumption: $\beta_0 = X^+ X \beta_0$

Penalised least squares criterion

= Ridge regression (RR) objective

$$\hat{\beta}_{\lambda}^{\text{RR}} := \operatorname{argmin}_{\beta \in \mathbb{R}^p} \mathcal{E}_{\lambda}(\beta) \quad \text{with} \quad \mathcal{E}_{\lambda}(\beta) := \frac{1}{2n} \|y - X\beta\|^2 + \frac{\lambda}{2} \|\beta\|^2, \quad \lambda \geq 0$$

- $\lambda = 0$: $\hat{\beta}_0^{\text{RR}} = X^+y$ minimum-norm solution of $y = X\beta$, $\hat{\Sigma}_0^{-1} := \hat{\Sigma}^+$ [Ali et al. 2019]
- $\lambda > 0$: $\hat{\beta}_{\lambda}^{\text{RR}} = \hat{\Sigma}_{\lambda}^{-1} \cdot n^{-1}X^{\top}y$, where $\hat{\Sigma}_{\lambda} := \hat{\Sigma} + \lambda I_p$ with $\hat{\Sigma} := n^{-1}X^{\top}X$
- eigenvalues of $\hat{\Sigma}$: $s_1 \geq \dots \geq s_p \geq 0$, eigenvectors: $v_1, \dots, v_p \in \mathbb{R}^p$
- for simplicity: assume $\hat{\Sigma}_{\lambda}$ has p distinct eigenvalues, $\langle X^{\top}y, v_i \rangle^2 > 0, i = 1, \dots, p$

Penalised least squares criterion

= Ridge regression (RR) objective

$$\begin{aligned}\mathcal{E}_\lambda(\beta) &= \frac{1}{2n} \|y - X\beta\|^2 + \frac{\lambda}{2} \|\beta\|^2 \\ &= \frac{1}{2} \|\hat{\Sigma}_\lambda^{1/2} \beta - y_\lambda\|^2 + \frac{1}{2n} \|y\|^2 - \frac{1}{2} \|y_\lambda\|^2\end{aligned}$$

with

$$y_\lambda := \frac{1}{n} \hat{\Sigma}_\lambda^{-1/2} X^\top y = \hat{\Sigma}_\lambda^{1/2} \beta_\lambda + \varepsilon_\lambda, \quad \beta_\lambda := \hat{\Sigma}_\lambda^{-1} \hat{\Sigma} \beta_0, \quad \varepsilon_\lambda := \frac{1}{n} \hat{\Sigma}_\lambda^{-1/2} X^\top \varepsilon$$

normal equations: $\hat{\Sigma}_\lambda \hat{\beta}_\lambda^{\text{RR}} = \hat{\Sigma}_\lambda^{1/2} y_\lambda \quad \text{s.t.} \quad \hat{\beta}_\lambda^{\text{RR}} = \beta_\lambda + \hat{\Sigma}_\lambda^{-1/2} \varepsilon_\lambda$

How to calculate the RR estimator in practice?



API Reference > sklearn.linear_model > Ridge

Ridge

```
class sklearn.linear_model.Ridge(alpha=1.0, *,  
    fit_intercept=True, copy_X=True, max_iter=None,  
    tol=0.0001, solver='auto', positive=False,  
    random_state=None)
```

[\[source\]](#)

Linear least squares with l2 regularization.

Minimizes the objective function:

$$\|y - Xw\|_2^2 + \alpha \|w\|_2^2$$

solver : {'auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga', 'lbfgs'}, default='auto'

Solver to use in the computational routines:

- 'auto' chooses the solver automatically based on the type of data.
- 'svd' uses a Singular Value Decomposition of X to compute the Ridge coefficients. It is the most stable solver, in particular more stable for singular matrices than 'cholesky' at the cost of being slower.
- 'cholesky' uses the standard `scipy.linalg.solve` function to obtain a closed-form solution.
- 'sparse_cg' uses the conjugate gradient solver as found in `scipy.sparse.linalg.cg`. As an iterative algorithm, this solver is more appropriate than 'cholesky' for large-scale data (possibility to set `tol` and `max_iter`).
- 'lsqr' uses the dedicated regularized least-squares routine `scipy.sparse.linalg.lsqr`. It is the fastest and uses an iterative procedure.
- 'sag' uses a Stochastic Average Gradient descent, and 'saga' uses its improved, unbiased version named SAGA. Both methods also use an iterative procedure, and are often faster than other solvers when both `n_samples` and `n_features` are large. Note that 'sag' and 'saga' fast convergence is only guaranteed on features with approximately the same scale. You can preprocess the data with a scaler from `sklearn.preprocessing`.
- 'lbfgs' uses L-BFGS-B algorithm implemented in `scipy.optimize.minimize`. It can be used only when `positive` is True.

How to calculate the RR estimator in practice?



API Reference > sklearn.linear_model > Ridge

Ridge

```
class sklearn.linear_model.Ridge(alpha=1.0, *,  
    fit_intercept=True, copy_X=True, max_iter=None,  
    tol=0.0001, solver='auto', positive=False,  
    random_state=None)
```

[\[source\]](#)

Linear least squares with L2 regularization.

Minimizes the objective function:

```
||y - Xw||^2_2 + alpha * ||w||^2_2
```

solver : {'auto', 'svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga', 'lbfgs'}, default='auto'

Solver to use in the computational routines:

- 'auto' chooses the solver automatically based on the type of data.
 - 'svd' uses a Singular Value Decomposition of X to compute the Ridge coefficients. It is the most stable solver, in particular more stable for singular matrices than 'cholesky' at the cost of being slower.
 - 'cholesky' uses the standard `scipy.linalg.solve` function to obtain a closed-form solution.
 - 'sparse_cg' uses the conjugate gradient solver as found in `scipy.sparse.linalg.cg`. As an iterative algorithm, this solver is more appropriate than 'cholesky' for large-scale data (possibility to set `tol` and `max_iter`).
 - 'lsqr' uses the dedicated regularized least-squares routine `scipy.sparse.linalg.lsqr`. It is the fastest and uses an iterative procedure.
 - 'sag' uses a Stochastic Average Gradient descent, and 'saga' uses its improved, unbiased version named SAGA. Both methods also use an iterative procedure, and are often faster than other solvers when both `n_samples` and `n_features` are large. Note that 'sag' and 'saga' fast convergence is only guaranteed on features with approximately the same scale. You can preprocess the data with a scaler from `sklearn.preprocessing`.
 - 'lbfgs' uses
- used only w

By iterative solvers!

Gradient descent/flow (GD/GF)

$$\eta > 0, k \in \mathbb{N}, \hat{\beta}_{\lambda,\eta,0}^{\text{GD}} = 0,$$

$$\begin{aligned}\hat{\beta}_{\lambda,\eta,k}^{\text{GD}} &:= \hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}} - \eta \nabla \mathcal{E}_{\lambda}(\hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}}) \\ &= \hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}} - \eta(\hat{\Sigma}_{\lambda} \hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}} - \hat{\Sigma}_{\lambda}^{1/2} y_{\lambda})\end{aligned}$$

Gradient descent/flow (GD/GF)

$$\eta > 0, k \in \mathbb{N}, \hat{\beta}_{\lambda, \eta, 0}^{\text{GD}} = 0,$$

$$\hat{\beta}_{\lambda, \eta, k}^{\text{GD}} := \hat{\beta}_{\lambda, \eta, k-1}^{\text{GD}} - \eta \nabla \mathcal{E}_{\lambda}(\hat{\beta}_{\lambda, \eta, k-1}^{\text{GD}})$$

$\eta_k = t/k,$
 $k \rightarrow \infty,$
 $t > 0$


$$= \hat{\beta}_{\lambda, \eta, k-1}^{\text{GD}} - \eta(\hat{\Sigma}_{\lambda} \hat{\beta}_{\lambda, \eta, k-1}^{\text{GD}} - \hat{\Sigma}_{\lambda}^{1/2} y_{\lambda})$$

Gradient descent/flow (GD/GF)

$$\eta > 0, k \in \mathbb{N}, \hat{\beta}_{\lambda,\eta,0}^{\text{GD}} = 0,$$

$$\hat{\beta}_{\lambda,\eta,k}^{\text{GD}} := \hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}} - \eta \nabla \mathcal{E}_{\lambda}(\hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}})$$

$\eta_k = t/k,$
 $k \rightarrow \infty,$
 $t > 0$


$$= \hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}} - \eta(\hat{\Sigma}_{\lambda} \hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}} - \hat{\Sigma}_{\lambda}^{1/2} y_{\lambda})$$

$$\hat{\beta}_{\lambda,t}^{\text{GF}} \text{ satisfying } \hat{\beta}_{\lambda,0}^{\text{GF}} = 0,$$

$$\frac{d}{dt} \hat{\beta}_{\lambda,t}^{\text{GF}} = -(\hat{\Sigma}_{\lambda} \hat{\beta}_{\lambda,t}^{\text{GF}} - \hat{\Sigma}_{\lambda}^{1/2} y_{\lambda}), t \geq 0$$

Gradient descent/flow (GD/GF)

$$\eta > 0, k \in \mathbb{N}, \hat{\beta}_{\lambda, \eta, 0}^{\text{GD}} = 0,$$

$$\hat{\beta}_{\lambda, \eta, k}^{\text{GD}} := \hat{\beta}_{\lambda, \eta, k-1}^{\text{GD}} - \eta \nabla \mathcal{E}_{\lambda}(\hat{\beta}_{\lambda, \eta, k-1}^{\text{GD}})$$

$$\begin{array}{l} \eta_k = t/k, \\ k \rightarrow \infty, \\ t > 0 \end{array} \downarrow = \hat{\beta}_{\lambda, \eta, k-1}^{\text{GD}} - \eta(\hat{\Sigma}_{\lambda} \hat{\beta}_{\lambda, \eta, k-1}^{\text{GD}} - \hat{\Sigma}_{\lambda}^{1/2} y_{\lambda})$$

$$\hat{\beta}_{\lambda, t}^{\text{GF}} = \hat{\Sigma}_{\lambda}^{-1/2} (I_p - R_t^{\text{GF}}(\hat{\Sigma}_{\lambda})) y_{\lambda},$$

$$R_t^{\text{GF}}(x) = \exp(-tx)$$

Gradient descent/flow (GD/GF)

$$\eta > 0, k \in \mathbb{N}, \hat{\beta}_{\lambda,\eta,0}^{\text{GD}} = 0,$$

$$\hat{\beta}_{\lambda,\eta,k}^{\text{GD}} := \hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}} - \eta \nabla \mathcal{E}_{\lambda}(\hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}})$$

$$\begin{array}{l} \eta_k = t/k, \\ k \rightarrow \infty, \\ t > 0 \end{array} \downarrow = \hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}} - \eta(\hat{\Sigma}_{\lambda} \hat{\beta}_{\lambda,\eta,k-1}^{\text{GD}} - \hat{\Sigma}_{\lambda}^{1/2} y_{\lambda})$$

$$\hat{\beta}_{\lambda,t}^{\text{GF}} = \hat{\Sigma}_{\lambda}^{-1/2} (I_p - R_t^{\text{GF}}(\hat{\Sigma}_{\lambda})) y_{\lambda},$$

$$R_t^{\text{GF}}(x) = \exp(-tx)$$

Conjugate gradients (CG)

Algorithm 1 Penalised CG method

- 1: $\hat{\beta}_{\lambda,0}^{\text{CG}} \leftarrow 0, q_0 \leftarrow \frac{1}{n} X^{\top} y, d_0 \leftarrow q_0, e_0 \leftarrow X d_0, k = 1$
 - 2: **while** (no division by zero) **do**
 - 3: $a_k \leftarrow \|q_{k-1}\|^2 / (\frac{1}{n} \|e_{k-1}\|^2 + \lambda \|d_{k-1}\|^2)$
 - 4: $\hat{\beta}_{\lambda,k}^{\text{CG}} \leftarrow \hat{\beta}_{\lambda,k-1}^{\text{CG}} + a_k d_{k-1}$
 - 5: $q_k \leftarrow q_{k-1} - \frac{a_k}{n} X^{\top} e_{k-1} - \lambda a_k d_{k-1}$
 - 6: $b_k \leftarrow \|q_k\|^2 / \|q_{k-1}\|^2$
 - 7: $d_k \leftarrow q_k + b_k d_{k-1}$
 - 8: $e_k \leftarrow X d_k$
 - 9: $k \leftarrow k + 1$
 - 10: **end while**
-

Gradient descent/flow (GD/GF)

Conjugate gradients (CG)

$$\hat{\beta}_{\lambda,t}^{\text{GF}} = \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R_t^{\text{GF}}(\hat{\Sigma}_{\lambda}))y_{\lambda},$$

$$R_t^{\text{GF}}(x) = \exp(-tx)$$

$$\hat{\beta}_{\lambda,k}^{\text{CG}} = \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R_k^{\text{CG}}(\hat{\Sigma}_{\lambda}))y_{\lambda},$$

$$R_k^{\text{CG}} := \arg \min_{P_k} \|P_k(\hat{\Sigma}_{\lambda})y_{\lambda}\|^2$$

[Phatak and de Hoog 2002]

Gradient descent/flow (GD/GF)

$$\hat{\beta}_{\lambda,t}^{\text{GF}} = \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R_t^{\text{GF}}(\hat{\Sigma}_{\lambda}))y_{\lambda},$$
$$R_t^{\text{GF}}(x) = \exp(-tx)$$

Conjugate gradients (CG)

$$\hat{\beta}_{\lambda,k}^{\text{CG}} = \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R_k^{\text{CG}}(\hat{\Sigma}_{\lambda}))y_{\lambda},$$

$$R_k^{\text{CG}} := \arg \min_{P_k} \|P_k(\hat{\Sigma}_{\lambda})y_{\lambda}\|^2$$

[Phatak and de Hoog 2002]

$$\Rightarrow \hat{\beta}_{\lambda,p}^{\text{CG}} = \hat{\beta}_{\lambda}^{\text{RR}}$$

Gradient descent/flow (GD/GF)

Conjugate gradients (CG)

$$\hat{\beta}_{\lambda,t}^{\text{GF}} = \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R_t^{\text{GF}}(\hat{\Sigma}_{\lambda}))y_{\lambda},$$
$$R_t^{\text{GF}}(x) = \exp(-tx)$$

continuous iteration path:

$$t = k + \alpha, k \in \llbracket 0, p - 1 \rrbracket, \alpha \in (0, 1],$$

$$\hat{\beta}_{\lambda,t}^{\text{CG}} := \hat{\Sigma}_{\lambda}^{-1/2}(I_p - R_t^{\text{CG}}(\hat{\Sigma}_{\lambda}))y_{\lambda},$$

$$R_t^{\text{CG}} := (1 - \alpha)R_k^{\text{CG}} + \alpha R_{k+1}^{\text{CG}}$$

[H. and Reiß 2025]

Error analysis

For in-sample prediction errors

- pure prediction error: $n^{-1} \|X(\hat{\beta} - \beta_0)\|^2 = \|\hat{\Sigma}^{1/2}(\hat{\beta} - \beta_0)\|^2$
- population risk:
 $\beta \mapsto \mathbb{E}[\mathcal{E}_\lambda(\beta) | X] = \frac{1}{2} \|\hat{\Sigma}_\lambda^{1/2}(\beta - \beta_\lambda)\|^2 + \frac{1}{2} \sigma^2 + \frac{1}{2} \|\hat{\Sigma}^{1/2} \beta_0\|^2 - \frac{1}{2} \|\hat{\Sigma}_\lambda^{1/2} \beta_\lambda\|^2$

Error analysis

For in-sample prediction errors

- pure prediction error: $n^{-1} \|X(\hat{\beta} - \beta_0)\|^2 = \|\hat{\Sigma}^{1/2}(\hat{\beta} - \beta_0)\|^2$
- population risk:
 $\beta \mapsto \mathbb{E}[\mathcal{E}_\lambda(\beta) | X] = \frac{1}{2} \|\hat{\Sigma}_\lambda^{1/2}(\beta - \beta_\lambda)\|^2 + \frac{1}{2} \sigma^2 + \frac{1}{2} \|\hat{\Sigma}^{1/2} \beta_0\|^2 - \frac{1}{2} \|\hat{\Sigma}_\lambda^{1/2} \beta_\lambda\|^2$
- excess penalised prediction error:
 $\|\hat{\Sigma}_\lambda^{1/2}(\hat{\beta} - \beta_\lambda)\|^2 = n^{-1} \|X(\hat{\beta} - \beta_\lambda)\|^2 + \lambda \|\hat{\beta} - \beta_\lambda\|^2 =: \ell_{\lambda, \beta_\lambda}^{\text{in}}(\hat{\beta})$
- regularised in-sample prediction loss:
 $\ell_{\lambda, \gamma}^{\text{in}}(\hat{\beta}) := \|\hat{\Sigma}_\lambda^{1/2}(\hat{\beta} - \gamma)\|^2 = n^{-1} \|X(\hat{\beta} - \gamma)\|^2 + \lambda \|\hat{\beta} - \gamma\|^2$
- prediction risk: $\mathcal{R}_{\lambda, \gamma}^{\text{in}}(\hat{\beta}) := \mathbb{E}[\ell_{\lambda, \gamma}^{\text{in}}(\hat{\beta}) | X]$

Standard prediction error decomposition

(Useful for linear regularisation methods)

Consider for any **residual filter** $R : [0, \infty) \rightarrow \mathbb{R}$ the estimator $\hat{\beta} = \hat{\Sigma}_\lambda^{-1/2}(I_p - R(\hat{\Sigma}_\lambda))y_\lambda$.

Then the **penalised prediction loss** satisfies $\ell_{\lambda, \gamma}^{\text{in}}(\hat{\beta}) = A_{\lambda, \gamma}(R) + S_\lambda(R) - 2C_{\lambda, \gamma}(R)$

with **approximation**, **stochastic** and **cross term errors**

$$A_{\lambda, \gamma}(R) := \|\hat{\Sigma}_\lambda^{1/2}(R(\hat{\Sigma}_\lambda)\beta_\lambda + (\gamma - \beta_\lambda))\|^2, \quad S_\lambda(R) := \|(I_p - R(\hat{\Sigma}_\lambda))\varepsilon_\lambda\|^2,$$

$$C_{\lambda, \gamma}(R) := \langle \hat{\Sigma}_\lambda^{1/2}(R(\hat{\Sigma}_\lambda)\beta_\lambda + (\gamma - \beta_\lambda)), (I_p - R(\hat{\Sigma}_\lambda))\varepsilon_\lambda \rangle.$$

If R is **deterministic**, then $\mathcal{R}_{\lambda, \gamma}^{\text{in}}(\hat{\beta}) = A_{\lambda, \gamma}(R) + \frac{\sigma^2}{n} \text{trace}((I_p - R(\hat{\Sigma}_\lambda))^2 \hat{\Sigma}_\lambda^{-1} \hat{\Sigma})$.

Prediction error decomposition for CG

[H. and Reiß 2025, Proposition 5.2]

- denote by $x_{1,t}$ the **smallest zero** on $[0, \infty)$ of R_t^{CG} , $t \in (0, p]$
- for $x \geq 0$, $R_{t,<}^{\text{CG}}(x) := R_t^{\text{CG}}(x)\mathbf{1}(x < x_{1,t})$, $R_{t,>}^{\text{CG}}(x) := R_t^{\text{CG}}(x)\mathbf{1}(x > x_{1,t})$

At $t \in [0, p]$, the **penalised CG estimator** satisfies for $\gamma = \beta_\lambda$

$$\ell_{\lambda, \beta_\lambda}^{\text{in}}(\hat{\beta}_{\lambda, t}^{\text{CG}}) = A_{\lambda, \beta_\lambda, t}^{\text{CG}} + S_{\lambda, t}^{\text{CG}} - 2C_{\lambda, \beta_\lambda, t}^{\text{CG}} \leq 2A_{\lambda, \beta_\lambda, t}^{\text{CG}} + 2S_{\lambda, t}^{\text{CG}}$$

with **approximation**, **stochastic** and **cross term errors** given by

$$A_{\lambda, \beta_\lambda, t}^{\text{CG}} := \|\hat{\Sigma}_\lambda^{1/2} R_{t,<}^{\text{CG}}(\hat{\Sigma}_\lambda)^{1/2} \beta_\lambda\|^2 + \|R_t^{\text{CG}}(\hat{\Sigma}_\lambda) y_\lambda\|^2 - \|R_{t,<}^{\text{CG}}(\hat{\Sigma}_\lambda)^{1/2} y_\lambda\|^2 \leq \|\hat{\Sigma}_\lambda^{1/2} R_{t,<}^{\text{CG}}(\hat{\Sigma}_\lambda)^{1/2} \beta_\lambda\|^2,$$

$$S_{\lambda, t}^{\text{CG}} := \|(I_p - R_{t,<}^{\text{CG}}(\hat{\Sigma}_\lambda))^{1/2} \varepsilon_\lambda\|^2, \quad C_{\lambda, \beta_\lambda, t}^{\text{CG}} := \langle R_{t,>}^{\text{CG}}(\hat{\Sigma}_\lambda) y_\lambda, \varepsilon_\lambda \rangle.$$

$$\rho_t := |(R_t^{\text{CG}})'(0)|,$$

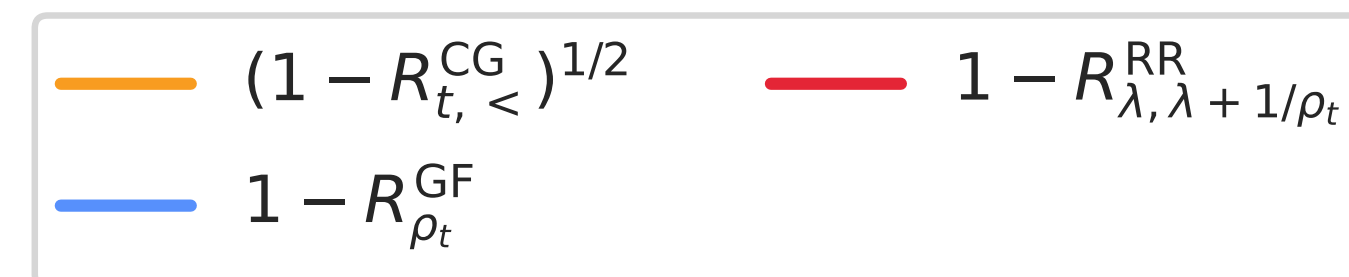
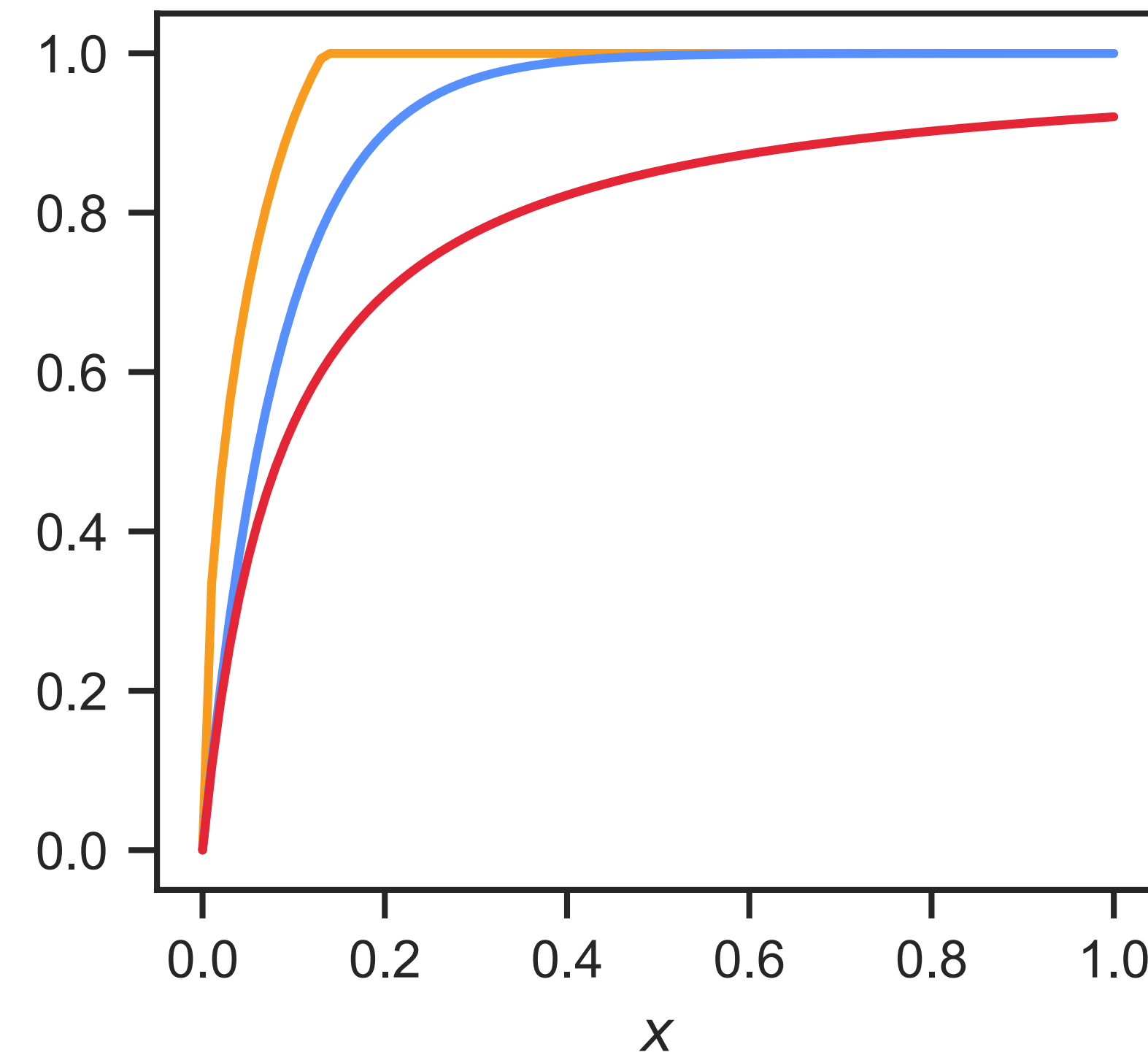
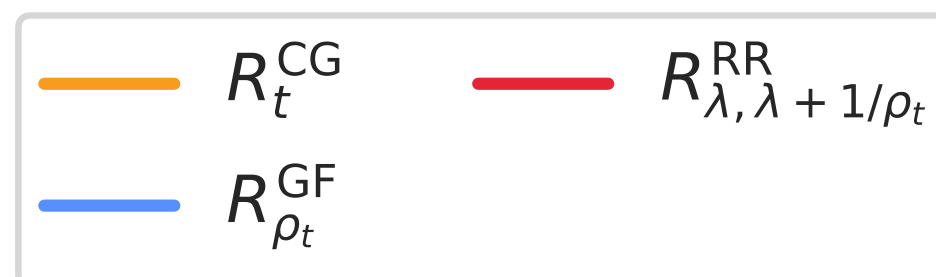
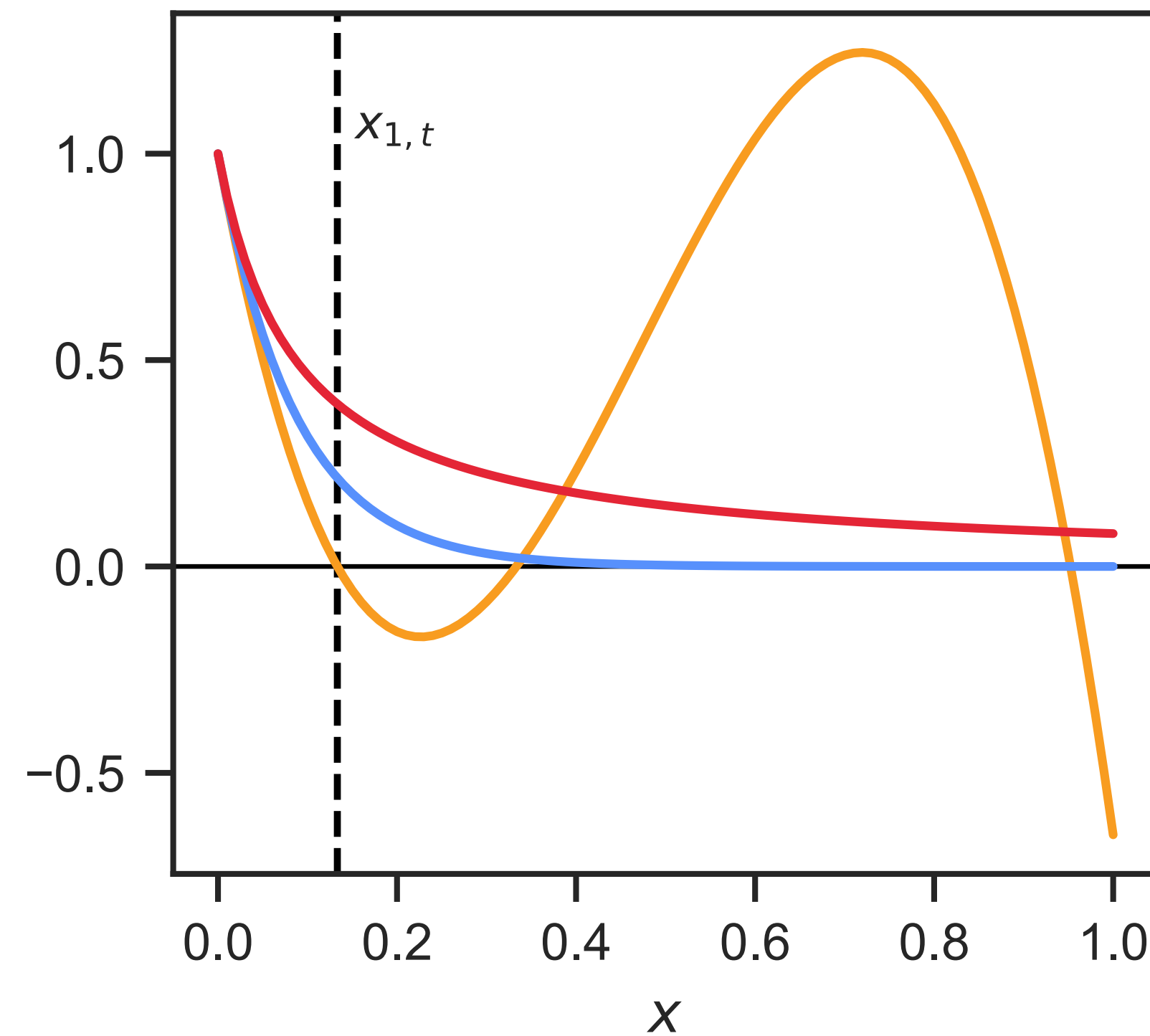
$$(1 - \rho_t x)_+ \leq R_t^{\text{CG}}(x) \leq \exp(-\rho_t)x, \quad x \in [0, x_{1,t}]$$

[H. and Reiß 2025, Lemma 4.7]

$$\rho_t := |(R_t^{\text{CG}})'(0)|,$$

$$(1 - \rho_t x)_+ \leq R_t^{\text{CG}}(x) \leq \exp(-\rho_t)x, \quad x \in [0, x_{1,t}]$$

[H. and Reiß 2025, Lemma 4.7]



$$\rho_t := |(R_t^{\text{CG}})'(0)|,$$

$$(1 - \rho_t x)_+ \leq R_t^{\text{CG}}(x) \leq \exp(-\rho_t)x, \quad x \in [0, x_{1,t}]$$

[H. and Reiß 2025, Lemma 4.7]

Let $t \in [0, p]$. Assume that $\gamma \in \mathbb{R}^p$ satisfies the condition (CROSS)

$$\langle \hat{\Sigma}_\lambda \exp(-\rho_t \hat{\Sigma}_\lambda / 2) \beta_\lambda, \gamma - \beta_\lambda \rangle \geq 0.$$

Then the **regularised in-sample prediction loss for the CG estimator** is bounded by

$$\ell_{\lambda, \gamma}^{\text{in}}(\hat{\beta}_{\lambda, t}^{\text{CG}}) \leq 4\bar{A}_{\lambda, \gamma, t}^{\text{CG}} + 2\bar{S}_{\lambda, t}^{\text{CG}}$$

with **γ -dependent approximation** and **stochastic error bounds**

$$\bar{A}_{\lambda, \gamma, t}^{\text{CG}} := \|\hat{\Sigma}_\lambda^{1/2}(\beta_\lambda - \exp(-\rho_t \hat{\Sigma}_\lambda / 2)\beta_\lambda - \gamma)\|^2, \quad \bar{S}_{\lambda, t}^{\text{CG}} := \|(\rho_t \hat{\Sigma}_\lambda \wedge 1)^{1/2} \varepsilon_\lambda\|^2.$$

Main result

Compare CG and GF errors up to a random time shift

- inverting $t \mapsto \rho_t/2$ leads to the **random times**
 $\tau_t := \inf\{\tilde{t} \in [0, p] \mid |(R_{\tilde{t}}^{\text{CG}})'(0)| \geq 2t\} \wedge p, \quad t \geq 0$
- under (CROSS), $\mathcal{R}_{\lambda, \gamma}^{\text{in}}(\hat{\beta}_{\lambda, \tau_t}^{\text{CG}}) \leq 4A_{\lambda, \gamma}(R_t^{\text{GF}}) + \frac{4\sigma^2}{n} \text{trace}((2t \wedge \hat{\Sigma}_{\lambda}^{-1})\hat{\Sigma})$
- remains to compare stochastic errors of CG and GF

For $t \geq \frac{1}{2(s_1 + \lambda)}$ set $C_{t, \lambda} := \left(\sum_{j \geq i_t} s_j \right) \left(\sum_{j < i_t} \frac{(s_{i_t} + \lambda)s_j}{s_j + \lambda} + \sum_{j \geq i_t} \frac{(s_j + \lambda)s_j}{s_{i_t-1} + \lambda} \right)^{-1} \mathbf{1}\left(t < \frac{1}{2}(s_p + \lambda)^{-1}\right)$

with $i_t := \min\{j \mid s_j < (2t)^{-1} - \lambda\}$. Then $C_{t, \lambda} \leq C_{t, 0}$ holds, and under (CROSS) for γ we have

$$\forall t \geq \frac{1}{2\|\hat{\Sigma}_{\lambda}\|} : \quad \mathcal{R}_{\lambda, \gamma}^{\text{in}}(\hat{\beta}_{\lambda, \tau_t}^{\text{CG}}) \leq \frac{4(1 + C_{t, \lambda})}{(1 - e^{-1/2})^2} \mathcal{R}_{\lambda, \gamma}^{\text{in}}(\hat{\beta}_{\lambda, t}^{\text{GF}}).$$

Examples

Polynomial and geometric decay of the eigenvalues

For $t \geq \frac{1}{2(s_1 + \lambda)}$ set

$$C_{t,\lambda} := \left(\sum_{j \geq i_t} s_j \right) \left(\sum_{j < i_t} \frac{(s_{i_t} + \lambda)s_j}{s_j + \lambda} + \sum_{j \geq i_t} \frac{(s_j + \lambda)s_j}{s_{i_t-1} + \lambda} \right)^{-1} \mathbf{1} \left(t < \frac{1}{2}(s_p + \lambda)^{-1} \right)$$

with $i_t := \min\{j \mid s_j < (2t)^{-1} - \lambda\}$.

Suppose for some $C > 0$, $\alpha > 1$, $\forall 2 \leq i \leq j \leq p : s_j/s_i \leq C(j/i)^{-\alpha}$.

Then all $C_{t,\lambda}$ are uniformly bounded:

$$C_{t,\lambda} \leq C_{t,0} \leq \frac{C \sum_{j \geq i_t} (j/i_t)^{-\alpha}}{i_t - 1} \leq \frac{C(\alpha + 1)}{\alpha - 1}.$$

Simulated data

solid lines:

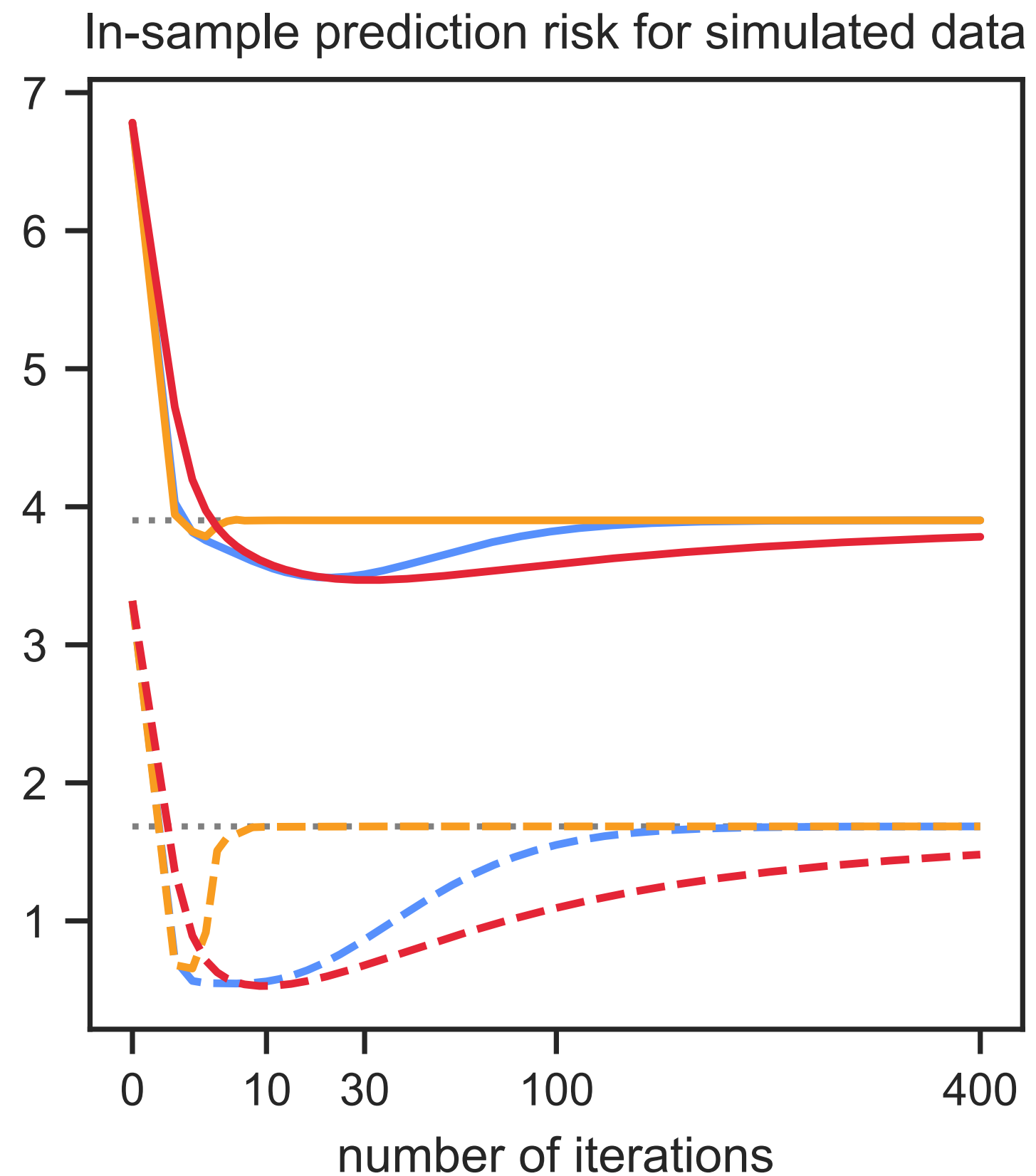
$$\gamma = \beta_0$$

dashed lines:

$$\gamma = \beta_\lambda$$

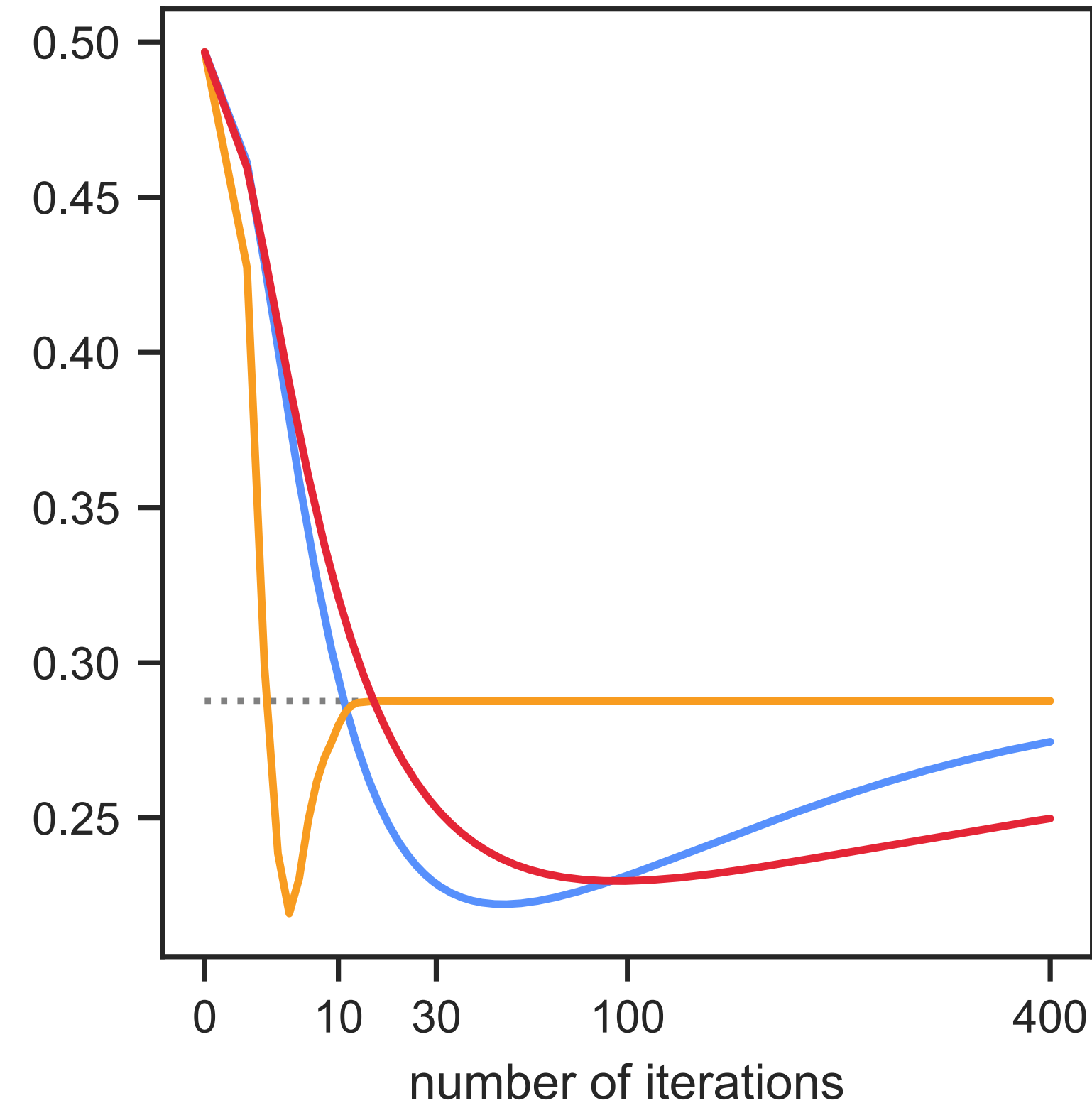
grey lines:

$$\hat{\beta}_\lambda^{\text{RR}}$$



GF CG RR

Out-of-sample criterion for Riboflavin data



Real data

solid lines:

$$\mathcal{E}_\lambda(\hat{\beta})$$

grey line:

$$\hat{\beta}_\lambda^{\text{RR}}$$

- $n = 400, p = 500, \lambda = 3$, 1000 Monte Carlo runs
- $\beta_0 \sim \mathcal{N}(0, p^{-1}I_p)$
- $x_i \sim \mathcal{N}(0, \Sigma), \lambda_1 = \dots = \lambda_{20} = 100, \lambda_{21} = \dots = \lambda_{500} = 1$
- $\varepsilon_i \sim \mathcal{N}(0, 6)$

- *Riboflavin* data from R package hdi
- $n = 71, p = 2n, \lambda = 0.1$
- 1000 splits: 50 training observations, 21 test data points
- ridge criterion $\mathcal{E}_\lambda(\hat{\beta})$ evaluated on test set

Electronic Journal of Statistics

Vol. 20 (2026) 425–444

ISSN: 1935-7524

<https://doi.org/10.1214/26-EJS2497>

Comparing regularisation paths of (conjugate) gradient estimators in ridge regression

Laura Hucker¹ , Markus Reiß¹ and Thomas Stark² 

¹*Institute of Mathematics, Humboldt-Universität zu Berlin, Germany,*
e-mail: huckerla@math.hu-berlin.de; mreiss@math.hu-berlin.de

²*Department of Statistics and Operations Research, University of Vienna, Austria,*
e-mail: thomas.stark@univie.ac.at

L. Hucker, M. Reiß, T. Stark (2026). Comparing regularisation paths of (conjugate) gradient estimators in ridge regression. *Electron. J. Stat.* 20(1), 425–444. doi:10.1214/26-EJS2497



L. Hucker, M. Reiß (2025). Early stopping for conjugate gradients in statistical inverse problems. *Numer. Math.* 157(5), 1739–1791. doi:10.1214/26-EJS2497

L. Hucker, M. Wahl (2023). A note on the prediction error of principal component regression in high dimensions. *Theory Probab. Math. Stat.* 109, 37–53. doi:10.1090/tpms/1196

**Thanks a lot for
your attention!**

Some further references

A. Ali, J.Z. Kolter, R.J. Tibshirani (2019). A continuous-time view of early stopping for least squares regression. In: K. Chaudhuri, M. Sugiyama (eds.) Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics. *Proceedings of Machine Learning Research*, vol. 89, pp. 1370–1378

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, É. Duchesnay (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* 12(85), 2825–2830

A. Phatak, F. de Hoog (2002). Exploiting the connection between PLS, Lanczos methods and conjugate gradients: alternative proofs of some properties of PLS. *J. Chemom.* 16(7), 361–367. doi:10.1002/cem.728