JOURNAL OF REPRODUCIBLE STATISTICS
FAIR Press Journals · jrepstat.fairpressjournals.com
Vol. 1 (2026), Art. 10
10.68365/jrepstat.2026.10

RESEARCH ARTICLE · SIMULATION STUDY

The Constrained LASSO is a Portfolio: A numerical check of the CLA–LASSO identity under constraints

Thomas Schmelzer*

Jebel Quant Research, Abu Dhabi
*Corresponding author: thomas.schmelzer@gmail.com

ARTICLE INFO

Keywords
constrained LASSO
critical line algorithm
solution path
compositional data
portfolio optimisation
reproducibility

History
Received 25 Oct 2026
Final version 26 Oct 2026
Accepted 26 Oct 2026
Version 1

Handling editor
Matthias Templ

ABSTRACT

Schmelzer and Hastie showed that the solution path of the constrained LASSO and the path of Markowitz's Critical Line Algorithm (CLA) are the same curve. This note checks that identity numerically. It poses a constrained regression as a portfolio and hands it to a released CLA written for portfolios. On caps, signs, equality and inequality rows, monotone fits and zero sums, on simulated and real data, up to 500 predictors and on problems built to break a path algorithm, the paths reproduce LARS, an interior-point solver and three independent path algorithms, and meet the LASSO's optimality conditions to rounding, or to 10^{-9} where a path with more predictors than observations nears an interpolating fit. A design with two nearly identical columns is declined with an error rather than answered wrongly. With more predictors than observations the paths stay exact, where the path algorithms for general constraints add a ridge that moves the end of the path.

✓ Reproduced verified 2026-10-26 REPRODUCIBILITY RECORD

Result	Reported results reproduced. All counts, declines and certificates reproduce on macOS arm64 and Linux. Near interpolating fits two residual bounds are exceeded, one of them 14-fold.	✓ *measured*
Entry point	`cd code/lasso-327bcd7a798d51e8a92ac6babdfb8ede62ef1e42/preprints/constrained_lasso && make reproduce`	✓ *executed*
Environment	Python 3.12 from the lockfiles via uv, Julia 1.12.6, R 4.6.1 with Rglpk 0.6.5.1, Linux aarch64 container and macOS arm64	✓ *measured*
Determinism	Seeded.	*declared*
Archive	`the-constrained-lasso-is-a-portfolio-a-numerical-check-of-the-cla-lasso-identity-under-constraints-v1-repro.zip` · reviewed bundle SHA-256 `17fde9d45a841795b52d55cc99ce5585bfde3b78d31a540d4bfa2a6aea389552`	✓ *computed*
Code	`https://github.com/Jebel-Quant/lasso/tree/main/preprints/constrained_lasso`	*declared*
Data	Open.	*declared, route checked*

1 A constrained regression

Tibshirani (1996) introduced the LASSO as least squares with a single constraint, a budget on the ℓ_1 norm of the coefficients:

CITE AS Schmelzer, T. (2026). The Constrained LASSO is a Portfolio: A numerical check of the CLA–LASSO identity under constraints. Journal of Reproducible Statistics, 1, 10. https://doi.org/10.68365/jrepstat.2026.10

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 \quad \text{s.t.} \quad \|\beta\|_1 \le t, \qquad \text{or equivalently} \qquad \min_{\beta} \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1.$$

LARS (Efron et al., 2004) and the homotopy method (Osborne et al., 2000) compute the solution for every t, or every λ, at once. The solution is piecewise linear in the parameter, so the whole *path* is a finite list of *knots*. scikit-learn's `lars_path` (Pedregosa et al., 2011) takes X, y and one option, `positive`, for nonnegative coefficients. There is no place for any other constraint.

Many regressions need one. The diabetes data that LARS was introduced with (Efron et al., 2004) illustrate what a constraint can do: $n = 442$ patients, ten standardised baseline variables and, as the response, a measure of disease progression a year later. Six of the variables are blood serum measurements, and three of them are closely related measures of cholesterol: total cholesterol (`s1`), LDL (`s2`) and HDL (`s3`). Least squares plays them off against each other. It gives total cholesterol a coefficient of -792 and LDL one of 477: two large effects of opposite sign that mostly cancel. HDL gets 101, as if more of it made the disease progress faster. Low HDL is a risk factor for incident diabetes (Wilson et al., 2007); the response here is progression in patients who have the disease, so that finding suggests a negative sign but does not establish one. The LASSO path ends at the least-squares fit, so it gets there too. A cap on every coefficient, $|\beta_j| \le u$, stops the offsetting. We take $u = 396.1$, half the largest least-squares coefficient, a value chosen after seeing the fit. The example shows what a cap does to offsetting coefficients; it is not offered as a better model of the disease.

Figure 1 shows what the cap does. Without it the path passes the cap at ten of its 13 knots and ends with HDL positive. With it no coefficient passes the cap, and HDL stays negative along the whole path, although no sign is imposed. That does not hang on the choice of u: every whole-number cap up to 637 keeps HDL negative along the whole path, and only looser caps let it turn positive. That is a property of these data, not a finding about HDL. The capped path has 13 knots.

Both panels are exact paths. They come from the Critical Line Algorithm (CLA), which Markowitz (1956) wrote for portfolios, applied to one of the two encodings described below. The open-source package `cvxcla` (Schmelzer, 2026a) carries them out (Algorithm 1), and the right panel is one call:

```
from cvxcla import Lasso
knots = Lasso(X, y, lower=-u * np.ones(p), upper=u * np.ones(p)).path
```

The same call also takes signs (a bound of zero), equality rows `a, b` and inequality rows `g, h`, and picks the encoding that fits them.

What this note shows Schmelzer and Hastie (2026) prove that the CLA and the constrained LASSO (Gaines et al., 2018; James et al., 2020) trace the same curve; Table 1 gives the dictionary. Portfolio managers have always needed constraints, such as a budget, position limits and sector exposures, and

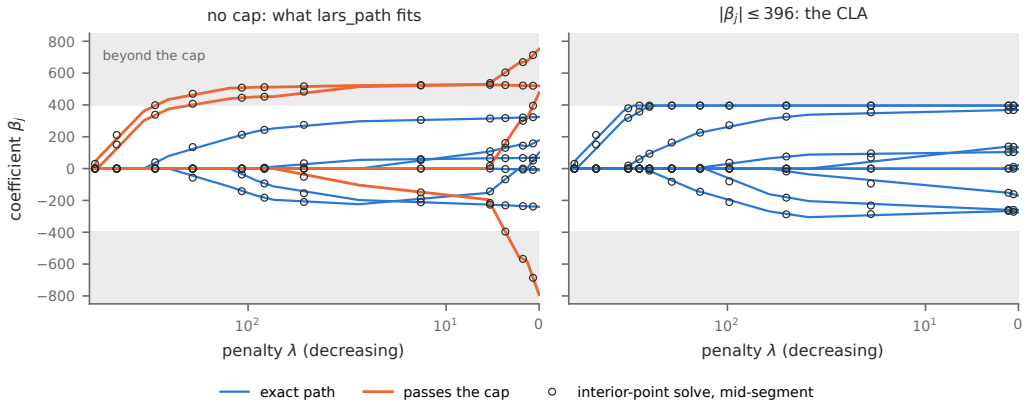


Figure 1 The opening example: the diabetes LASSO path without and with the cap $|\beta_j| \le u = 396.1$. The shaded bands are beyond the cap. Left: the plain LASSO, the path `lars_path` computes; the coefficients that pass the cap are in orange. Right: every coefficient capped. Lines: the exact path, by the CLA in the penalty form. Circles: an interior-point solve at the midpoint of every segment. The horizontal axis is λ, decreasing to the right, on a symmetric-log scale: logarithmic above $\lambda_{\max}/50$ and linear below it, so that the end of the path at $\lambda = 0$ is shown.

Markowitz built them into the CLA from the start. So a CLA written for portfolios, with a thin layer that poses the regression as a portfolio, should give exact constrained-LASSO paths. This note shows that it does, or stops with an error where its arithmetic cannot be trusted, and checks every path against references that know nothing of portfolios. The layer is two encodings, Propositions 1 and 2. The propositions need nothing from Schmelzer & Hastie (2026) beyond the classical equivalence of the bound and penalty forms of the LASSO; that paper supplies the general identity of which they are instances. The checks also exposed two defects in the CLA that refused ordinary portfolio problems too, a tie for the largest expected return and a covariance of low rank. Both were fixed in `cvxcla` 2.6.0 (Schmelzer, 2026a). Drawing the hostile problems of Section 5.4 over more seeds exposed three more, on a design whose columns are nearly collinear: two in the check that a bound held along a segment is still optimal, which refused a correct path, and one in the plain LASSO homotopy, which lost the second of two events at the same λ. All three were fixed in 2.7.1, the release used here. Brodie et al. (2009) put an ℓ_1 penalty on Markowitz portfolios to make them sparse, and a companion note (Schmelzer, 2026b) computes efficient frontiers with a LASSO solver. Both use regression to build portfolios. This note goes the other way. Section 2 recalls the CLA, Section 3 gives the two encodings, Section 4 says where the path starts and ends and collects them in Algorithm 1, Section 5 checks the paths, and Section 6 gives the limits.

Table 1 The dictionary between the two problems (Schmelzer & Hastie, 2026).

regression	portfolio
coefficient β_j	position in asset j
$X^\top X$	covariance of the asset returns
$X^\top y$	expected returns (penalty form: a benchmark)
$\|\beta\|_1$	gross exposure
penalty λ	risk tolerance γ (penalty form)
bound $\|\beta\|_1 \le t$	leverage cap t, $\gamma = 1/t$ (budget form)
$A\beta = b, l \le \beta \le u$	budget rows and position limits
knot of the LASSO path	turning point of the CLA

The *penalty form* (Section 3.1) keeps λ as the parameter and takes bounds, equality rows $A\beta = b$ and inequality rows $C\beta \le h$; it is the general form. The *budget form* (Section 3.2) keeps the ℓ_1 budget, as Markowitz's long-only, fully invested problem with a cash asset, and takes homogeneous constraints such as the zero sum $\mathbf{1}^\top\beta = 0$ of compositional data in one run.

Related work Exact paths of the constrained LASSO are not new. Gaines, Kim and Zhou (2018) derive a path algorithm from its optimality conditions that takes general linear rows. For $C\beta = 0$ alone, the Python package `c-lasso` (Simpson et al., 2021) has a path algorithm with a small ridge term added. For zero sums on blocks of coefficients, the comlasso algorithm of Jeon et al. (2020) traces the exact path, also with $p > n$. None was derived with portfolios in mind, so Section 5 runs all three as independent checks. Hobson, Butcher and Lentati (2026) revisit Gaines et al. with CLASSO. The linear system of Gaines et al. is singular whenever a sign constraint binds on a zero coefficient, so CLASSO takes the direction from a small quadratic program instead. In the CLA a sign or a cap is a bound, not a row, so that singularity does not arise. CLASSO adds a ridge when $n < p$, and its code had not been released when we wrote, so we do not run it. The CLA is an active-set method of the same kind, written for portfolios and used here as released. With $p > n$ the other path algorithms for general constraints add a ridge, which decides where their path ends; the CLA needs none (Section 4). Microbiome studies often have more taxa than samples, so this is the case a user of the zero-sum LASSO meets first. Table 6 collects what each accepts.

Notation The design is $X \in \mathbb{R}^{n\times p}$, with n observations and p predictors, and the response is $y \in \mathbb{R}^n$. Write $G = X^\top X$ and $c = X^\top y$. The constrained LASSO is

$$\mathrm{L}_\lambda : \quad \min_\beta \frac{1}{2}\|y - X\beta\|_2^2 + \lambda\|\beta\|_1 \quad \text{s.t.} \quad A\beta = b, \quad C\beta \le h, \quad l \le \beta \le u, \tag{1}$$

with $A \in \mathbb{R}^{m\times p}$, $C \in \mathbb{R}^{k\times p}$ and $l_j \le 0 \le u_j$, infinite entries allowed, and a nonempty feasible set. A bound that excludes zero, such as $\beta_j \ge \ell$ with $\ell > 0$, is written as the row $-\beta_j \le -\ell$. With no rows and infinite

bounds it is Tibshirani's LASSO. We assume, as (Schmelzer & Hastie, 2026) does, that L_λ has a unique minimiser $\beta(\lambda)$ for every $\lambda > 0$. Without constraints this holds when the columns of X are in general position (R. J. Tibshirani, 2013), and with constraints when X has full column rank (Hobson et al., 2026, Theorem 6). For constrained problems with $p > n$ we know of no such result. The checks of Section 5 test optimality, not uniqueness: where the minimiser is not unique, they confirm that the path is a path of minimisers.

2 The Critical Line Algorithm

The CLA starts with the portfolio of highest expected return and lowers the risk tolerance; between corners, the *turning points* or knots, the optimal portfolio moves along a straight line. Precisely, it solves the family of quadratic programs

$$\mathrm{P}_\gamma : \quad \min_x \frac{1}{2} x^\top \Sigma x - (q + \gamma\mu)^\top x \quad \text{s.t.} \quad Ax = b, \quad Cx \le h, \quad l \le x \le u, \tag{2}$$

for every γ from $+\infty$ down to 0, with $\Sigma \succcurlyeq 0$. Markowitz's algorithm (Markowitz, 1956) handles general linear constraints; we write them as equality rows, inequality rows and bounds, the form `cvxcla` takes. Here x is the vector of holdings, Σ the covariance, μ the expected returns and γ the risk tolerance. The term q is a benchmark. With $q = \Sigma\bar{x}$ the objective is, up to a constant, the tracking variance against a portfolio $\bar{x}$ less $\gamma\mu^\top x$: the same CLA after the bounds and right-hand side are shifted by $\bar{x}$. With $q = 0$ it is Markowitz's problem.

Segments Write $g = \Sigma x - q - \gamma\mu + A^\top\nu + C^\top\eta$ for the gradient of the Lagrangian. A point x with multipliers ν and η is optimal when $g_j = 0$ on the *free* coordinates $\mathcal{F} = \{j : l_j < x_j < u_j\}$, $g_j \ge 0$ where $x_j = l_j$, and $g_j \le 0$ where $x_j = u_j$. In addition $\eta \ge 0$, with $\eta_i = 0$ on every row that is slack, $C_i x < h_i$. Call the coordinates at a bound *blocked*, $\mathcal{B}$, and the rows held tight *active*, $\mathcal{R}$. An active row is held as an equality. In the system below it is stacked under A, with its right-hand side h_i under b and its multiplier η_i under ν; the slack rows take no part. To keep the display short, A, b and ν stand for these stacks. With $\mathcal{F}$, $\mathcal{B}$ and $\mathcal{R}$ fixed, $x_\mathcal{F}$ and ν solve the bordered system

$$\begin{bmatrix} \Sigma_{\mathcal{F}\mathcal{F}} & A_\mathcal{F}^\top \\ A_\mathcal{F} & 0 \end{bmatrix} \begin{bmatrix} x_\mathcal{F} \\ \nu \end{bmatrix} = \begin{bmatrix} q_\mathcal{F} - \Sigma_{\mathcal{F}\mathcal{B}} x_\mathcal{B} \\ b - A_\mathcal{B} x_\mathcal{B} \end{bmatrix} + \gamma \begin{bmatrix} \mu_\mathcal{F} \\ 0 \end{bmatrix}, \tag{3}$$

so $x_\mathcal{F} = \alpha + \gamma\delta$ and $\nu = \nu_0 + \gamma\nu_1$ are affine in γ. The gradient on the blocked coordinates is affine too, $g_\mathcal{B} = g_0 + \gamma g_1$. One factorisation and two right-hand sides give the whole segment.

Events Lowering γ, the segment ends at the first of four events (Schmelzer & Hastie, 2026, Section 2.1): a free coordinate reaches a bound, a blocked coordinate's gradient changes sign, a slack row becomes tight, or an active row's multiplier falls to zero. Each is the root of an affine function of γ, such as $\gamma = -g_{0j}/g_{1j}$. The coordinate or row concerned changes sides, (3) is solved again, and the walk continues; with no event left, the last segment runs to $\gamma = 0$.

The start As $\gamma \to \infty$ the linear term dominates, and the path starts at a vertex maximising $\mu^\top x$ over the feasible set. A linear program finds it. Suppose the maximiser is unique, and as many of its coordinates lie strictly inside their bounds as there are equality rows and rows tight at it. Then the bordered matrix in (3) is nonsingular and the walk can begin. This is the nondegenerate case; `cvxcla` declines a degenerate start.

3 Two encodings

3.1 *The penalty form*

The first encoding reads the penalty $\lambda\|\beta\|_1$ as a cost: every unit of gross position, long or short, costs λ units of return. To make that a linear cost, each coefficient is split into a long and a short position, both nonnegative. The constraints on β become constraints on the positions, unchanged in kind.

Formally, let S have a column e_j for every j with $u_j > 0$ and a column $-e_j$ for every j with $l_j < 0$, so that $\beta = Sz$ with $z \ge 0$. Let $\bar{u}$ collect the matching bounds u_j and $-l_j$. A coefficient whose bounds are both zero is fixed and can be dropped. Every j then has at least one column, and S has full row rank.

Proposition 1. (Penalty as cost). *For $\lambda > 0$, $\beta(\lambda) = Sz(\lambda)$, where $z(\lambda)$ solves the CLA instance P_γ with*

$$\Sigma = S^\top G S, \quad \mu = -\mathbf{1}, \quad q = S^\top c, \quad \text{rows } ASz = b, CSz \le h, \quad 0 \le z \le \bar{u}, \quad \gamma = \lambda.$$

Moreover q lies in the range of Σ, so the benchmark reading of Section 2 applies: the program tracks a least-squares fit and pays λ for every unit of gross position.

Proof. Up to the constant $\frac{1}{2}\|y\|^2$, the CLA objective at $\gamma = \lambda$ is $\frac{1}{2}\|y - XSz\|^2 + \lambda \mathbf{1}^\top z$. For a fixed β feasible in (1), the smallest $\mathbf{1}^\top z$ over $z \ge 0$ with $Sz = \beta$ and $z \le \bar{u}$ is $\|\beta\|_1$. The rows hold for every such z, since they constrain only $Sz = \beta$. It is attained by putting β_j on the column of its sign, which respects $\bar{u}$ because $l_j \le \beta_j \le u_j$. The minimum of the CLA program over z is therefore the minimum of L_λ over β. At a minimiser no coordinate holds both signs, since lowering both by the same amount keeps β and reduces the cost by 2λ times that amount. Hence $Sz(\lambda)$ minimises L_λ, and by uniqueness equals $\beta(\lambda)$. For the range claim, $c = X^\top y$ lies in the range of G, say $c = Gv$. Since S has full row rank, $v = S\bar{x}$ for some $\bar{x}$, and then $q = S^\top G S\bar{x} = \Sigma\bar{x}$, where $S\bar{x}$ is a least-squares fit. □

The covariance $S^\top G S$ is singular when both signs of a coefficient are allowed, but (3) is not. For $\lambda > 0$ the proof keeps the two halves of a coefficient out of $\mathcal{F}$ together. On the free coordinates Σ is then positive definite on the null space of the active rows, the standing assumption of Schmelzer & Hastie (2026, Section 2). The halves come apart only at $\lambda = 0$, which rounding can move to $\gamma \approx 10^{-16}$; `cvxcla` treats an event whose parameter is round-off as one at $\gamma = 0$.

Figure 2(a) shows the penalty form on the diabetes data (Efron et al., 2004), where there are no constraints and LARS can check every knot.

3.2 *The budget form*

With $A\beta = 0$ the penalty form starts degenerate (Section 4). The second encoding starts instead from the bound $\|\beta\|_1 \le t$, a cap t on the leverage, whose path is that of L_λ indexed by $t = \|\beta(\lambda)\|_1$ (Schmelzer & Hastie, 2026, Theorem 1), and divides by t. The split coefficients then add up to at most one, which is Markowitz's own setting. A cash asset with no risk and no return holds the rest.

Proposition 2. (Budget as simplex). *Suppose $b = 0$ and $l_j, u_j \in \{0, \pm\infty\}$, so that every constraint is homogeneous. Write $\beta = tSw$ with $w \ge 0$, and add a riskless asset $w_0 \ge 0$ with $\mathbf{1}^\top w + w_0 = 1$. Then for $t > 0$ the bound-form LASSO is the CLA instance*

$$\Sigma = \begin{bmatrix} S^\top G S & 0 \\ 0 & 0 \end{bmatrix}, \quad \mu = \begin{bmatrix} S^\top c \\ 0 \end{bmatrix}, \quad q = 0, \quad \gamma = 1/t,$$

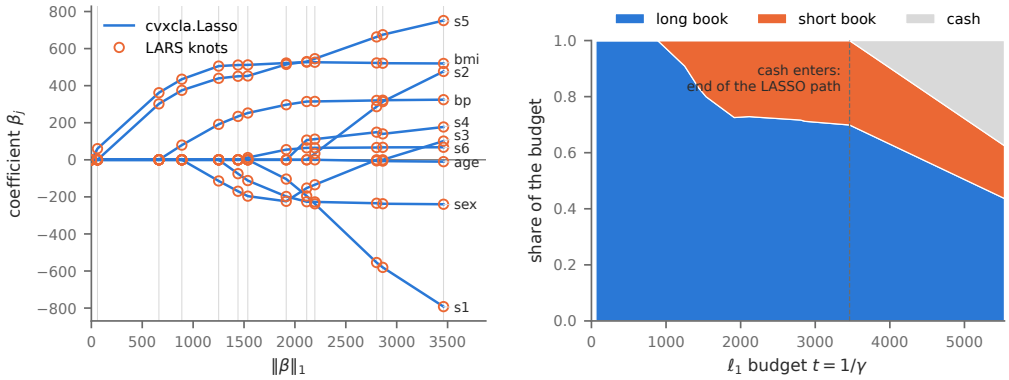


Figure 2 The diabetes LASSO path (Efron et al., 2004), both ways. *(a)* The CLA in the penalty form (lines) and the knots `lars_path` returns (Pedregosa et al., 2011) (circles); all 13 agree to 6.9×10^{-14}, relative. *(b)* The same path seen by the budget form as a portfolio: the long and short positions Sw and the cash weight w_0 share a budget of one, against the leverage $t = 1/\gamma$. Cash enters at $t = \|\beta(0)\|_1 = 3460$, the end of the LASSO path; to its right the least-squares fit is diluted by cash.

with rows $\mathbf{1}^\top w + w_0 = 1$ *and* $ASw = 0$ *and bounds* $w, w_0 \geq 0$. *This is Markowitz's long-only, fully invested problem with a cash asset of zero variance and zero return. If* η *is the multiplier of the budget row and* ν *those of* $ASw = 0$, *then* $\lambda = t\eta$ *and the LASSO's multipliers for* $A\beta = 0$ *are* $t\nu$.

Proof. As in Proposition 1, $\|\beta\|_1 \leq t$ is $\mathbf{1}^\top z \leq t$ for the split $z = tw$, and w_0 is its slack. The constraint $ASz = 0$ is unchanged by the scale t, which is where homogeneity is needed. A nonzero right-hand side would become $b/t = \gamma b$, a constraint that moves with the parameter. Substituting $z = tw$ gives the objective $t^2(\frac{1}{2}w^\top S^\top GSw - \gamma c^\top Sw)$ plus a constant, and dividing by $t^2 > 0$ gives P_γ. For the multipliers, multiply the CLA's stationarity row for a free coordinate, $\left(S^\top GSw\right)_j - \gamma\left(S^\top c\right)_j + \eta + \left(S^\top A^\top \nu\right)_j = 0$, by t. This gives $\left(S^\top(G\beta - c)\right)_j + t\eta + \left(S^\top A^\top t\nu\right)_j = 0$, the LASSO's stationarity condition with penalty $t\eta$ and constraint multipliers $t\nu$. The cash row reads $\eta \geq 0$ while cash is held at zero. □

The two forms give the same knots unless the support's sign vector lies in the span of the active constraint normals (Schmelzer & Hastie, 2026, Corollary 1 and Proposition 1). Under a zero sum that never happens: a nonzero β that sums to zero has coefficients of both signs.

4 Where the path starts and ends

The two encodings differ most at the ends of the path. A nondegenerate start lets the walk begin at once; a degenerate one needs a relaxation.

Nondegenerate starts In the penalty form without equality rows the starting vertex is $z = 0$: all positions are closed, because each costs more than it can earn. The first event is the first entry, at $\gamma = \max_j |c_j|$ over the allowed signs, which is LARS's $\lambda_{\max}$. Rows $C\beta \leq h$ with $h > 0$ are slack at $z = 0$, so they leave the start unchanged, and a row joins the active set when the path reaches it, like a bound. With equality rows, or a row with $h_r < 0$ that leaves $z = 0$ infeasible, the start is the linear program $\min \mathbf{1}^\top z$ over $ASz = b$, $CSz \leq h$, $0 \leq z \leq \bar{u}$: the point of smallest ℓ_1 norm on the constraint set. For generic A and $b \neq 0$ it is a unique vertex with m nonzeros, and the path holds still there, $\delta = 0$, until a second coefficient enters. In general, every constrained LASSO path is constant from some λ_0 on, at a least-squares fit over the feasible points of smallest ℓ_1 norm, and λ_0 takes the place of $\lambda_{\max}$ (Schmelzer & Hastie, 2026, Proposition 3). That fit can take a quadratic program to find. Under the fully invested constraint $\mathbf{1}^\top\beta = 1$ of sparse portfolio regression (Brodie et al., 2009) every long-only β in the simplex has $\|\beta\|_1 = 1$, so the starting linear program is not unique; `cvxcla` starts on the tied face where its quadratic term is smallest, which is that fit.

In the budget form the start, at $\gamma = \infty$ and $t = 0$, is the linear program $\max c^\top Sw$ over the simplex: put the whole budget on the best asset. Without equality rows that is the predictor most correlated with the response. It enters with the sign of its correlation and with $\lambda = \lim \eta/\gamma = \max_j |c_j|$, just as in LARS. Under $\mathbf{1}^\top\beta = 0$ it is one long and one short position of weight $\frac{1}{2}$ each, the most and the least correlated predictors, entering together. That is why two coefficients leave zero at once in Figure 3. Both starts are generically unique and nondegenerate.

Degenerate starts With $b = 0$ the penalty form's starting vertex is $z = 0$, with no free coordinates for $m > 0$ rows. This is a degenerate vertex: the multipliers at the start are not determined by the active set, and (3) is singular. For homogeneous equality rows with signs the budget form is the remedy, because rescaling to the simplex moves the start off the origin. It cannot take rows or caps, which move with the parameter once rescaled (Proposition 2). Caps or inequality rows under a zero sum therefore leave only the penalty form, which starts degenerate there. So do homogeneous rows ($h = 0$), such as the ordering constraints of a monotone fit: all are tight at $\beta = 0$.

`cvxcla` resolves both by relaxation, an implementation device rather than part of the method. It traces the problem twice, with the homogeneous rows relaxed to $h_r = \varepsilon$ and 2ε and a zero sum to $A\beta = \varepsilon Ad$ and $2\varepsilon Ad$ for a generic feasible point d. In release 2.7.1, ε is 10^{-6} times the scale $\|c\|_\infty / \max_j G_{jj}$ of a coefficient, then 10^{-7} and 10^{-8} times it if needed. Both runs start nondegenerate. The relaxation enters only the right-hand side of (3), so where the two runs pass through the same active sets their knots are affine in ε, and $2(\lambda_k, \beta_k)(\varepsilon) - (\lambda_k, \beta_k)(2\varepsilon)$ is the knot at $\varepsilon = 0$. That condition cannot be checked in advance. `cvxcla` therefore returns the extrapolated path only if a linear program finds multipliers that satisfy the

optimality conditions of (1) at every knot and segment midpoint, to 10^{-9} relative to $\|c\|_\infty$. Otherwise it retries with the next ε and, after the last, declines. The extrapolation only proposes a path; the certificate is what guarantees the path returned. The paths it returns are checked again, like every other path, in Section 5.

The end of the path As $\lambda \to 0^+$ the path ends at the least-squares fit of smallest ℓ_1 norm under the constraints (Schmelzer & Hastie, 2026, Proposition 3). In the budget form this is where cash leaves its bound, its gradient η reaching zero (Figure 2(b)), and the run stops there. With $p > n$ the end is an interpolating fit, and there that event wins a tie, since every correlation vanishes together with η. $X^\top X$ is then singular, but the bordered system needs a positive definite Hessian only on the null space of the active rows, the standing assumption of Schmelzer & Hastie (2026), so the free set can outgrow the rank of X. The other path algorithms for general constraints add a ridge: 10^{-4} in Gaines et al. when $n < p$, 10^{-3} in `c-lasso`, and $10^{-4} \max_j G_{jj}$ in CLASSO. At that ridge the authors of CLASSO report objectives up to 4×10^{-2}, relative, above the unregularised ones (Hobson et al., 2026, Table 3). Each then traces a nearby problem whose path ends elsewhere.

Algorithm 1: the constrained LASSO path from a CLA.

1. Form $G = X^\top X$, $c = X^\top y$ and the split S of Section 3.1.
2. If there are equality rows and every constraint is homogeneous ($b = 0$, $l_j, u_j \in \{0, \pm\infty\}$, no inequality rows), use the budget form of Proposition 2. Run the CLA from $\gamma = \infty$ and stop when cash leaves zero. A turning point at γ with budget multiplier η is the knot $\beta = Sw/\gamma$ at $\lambda = \eta/\gamma$.
3. Otherwise use the penalty form of Proposition 1 and run the CLA from $\gamma = \infty$ to 0. A turning point at γ is the knot $\beta = Sz$ at $\lambda = \gamma$. With $b \neq 0$, or a row with $h_r < 0$, the start is the feasible point of smallest ℓ_1 norm.
4. If the start is degenerate, trace the penalty form twice, with the homogeneous rows relaxed by ε and 2ε, extrapolate the knots to $\varepsilon = 0$, and return the path only if it passes the certificate.
5. Between two knots β is affine in λ, so any other point on the path is an interpolation.

`cvxcla.Lasso(X, y, a=A, b=b, g=C, h=h, lower=l, upper=u)` (release 2.7.1) carries out steps 1–5 and returns the knots. With no constraint at all it traces the plain LASSO by a homotopy instead.

5 Numerical evidence

Each check tests a prediction of Schmelzer & Hastie (2026). Their Lemma 1 makes L_λ and the CLA one parametric quadratic program, so the penalty form should return the knots of L_λ: LARS tests this knot by knot where it applies, and the interior-point solver and the optimality conditions everywhere. Their Theorem 1 and Corollary 1 give the bound form the same knots, so the budget form should return them too: comlasso and `c-lasso` test this on the zero sums.

5.1 *Test problems and checks*

Table 2 lists the test problems and their constraints. The diabetes problems use the data of Section 1, with the response centred, and *Box* is the opening example. *Nonnegative* carries the one constraint `lars_path` accepts. In *Cholesterol*, h is half the least-squares net effect of the three cholesterol measures, and the two rows bind at the end of the path. *Pinned* fits the first patient exactly, one nonhomogeneous equality row; its path holds still at its starting vertex until $\lambda = 4053$. *Wide* is a Gaussian design with 8 nonzero coefficients and noise of standard deviation 0.5. *Zero-sum* is a regression on compositional data in the style of Lin et al. (Lin et al., 2014), with the log-shares of p parts as predictors. A composition is known only up to scale, so only a zero sum makes the fit independent of the units. Five parts have nonzero true coefficients. The signs of *Zero-sum, signs* fall on the even-numbered parts, which the truth meets. The caps of *Zero-sum, caps* are at half the largest zero-sum least-squares coefficient. *Monotone* is a Gaussian design with $n = 100$, $p = 10$ and an ordered truth, under the rows $\beta_j - \beta_{j+1} \leq 0$. *Sum to one* predicts one unit's outcomes over 40 periods from those of 20 donors with weights that sum to one. Unlike the classic synthetic control (Abadie et al., 2010), which also requires the weights to be nonnegative, the weights here may be negative; adding the signs would be one more bound. *Floor*, not in the table, is the diabetes LASSO with $\beta_{\mathtt{bmi}}$ at least half its least-squares value, a row that leaves $\beta = 0$ infeasible. Its path has 13 knots. It

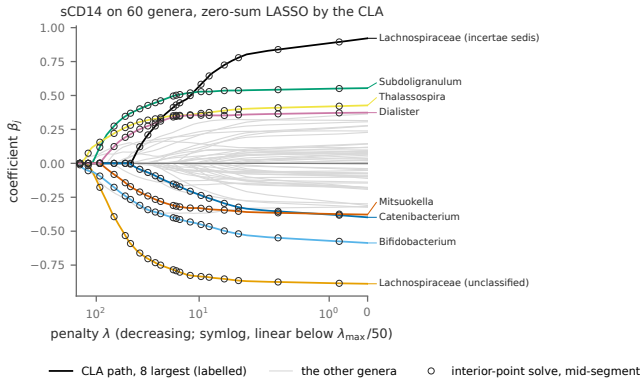


Figure 3 The zero-sum LASSO of sCD14 on the log-shares of 60 genera (Noguera-Julian et al., 2016): the path by the CLA in the budget form. The eight genera largest at $\lambda = 0$ are drawn in a colourblind-safe palette (Okabe & Ito, 2008) and labelled at their right-hand end, the others in grey, with an interior-point solve at every third segment midpoint for those eight (circles). Table 2 checks every midpoint of every coefficient. The axis is that of Figure 1: λ decreasing, symmetric-log, linear below $\lambda_{\max}/50$. A computational demonstration; no biological inference is drawn.

lies within 1.1×10^{-10} of an interior-point solve at every midpoint and meets the optimality conditions to 1.0×10^{-15}. *sCD14* is the same model on real data from a study of the gut microbiome in HIV infection (Noguera-Julian et al., 2016), as distributed with the R package `coda4microbiome` (Calle et al., 2023). The predictors come from the counts of 60 bacterial genera in 151 stool samples, and the response is the level of sCD14, a marker of inflammation, in thousands. Each count gets a pseudocount of one, since a zero has no logarithm, and the predictors are the centred log-shares. Its path has 60 knots (Figure 3). Like the diabetes problems, *sCD14* is a computational demonstration on real data. No biological inference is intended, and none should be drawn from its path.

Every path comes from `cvxcla.Lasso` 2.7.1. The bounds are passed even where they are infinite, so that the CLA computes it. Where LARS applies it is the reference, compared knot by knot in both λ and β. Elsewhere the reference is an interior-point solve (Clarabel through CVXPY, tolerances 10^{-12}) at the midpoint of every segment. There the path is affine in λ, so it is known exactly from its two ends. This comparison is only as sharp as the solver. Its largest gap in Table 2 is 4.8×10^{-8}, relative, on a path whose optimality conditions hold to rounding. The third check evaluates the optimality conditions of (1) directly at every knot and along every segment. Its unknowns are the multipliers κ of $A\beta = b$ and η of the active rows of $C\beta \le h$. A linear program chooses them, under $\eta \ge 0$, to minimise one number τ: the largest violation of all the conditions below, taken jointly. Each condition, written as an inequality $\phi_i(\kappa, \eta) \le 0$ that is affine in the multipliers, becomes the row $\phi_i(\kappa, \eta) \le \tau$, and the program minimises τ. Its optimum is zero exactly when β is optimal. The violation is then evaluated again, exactly, at the multipliers found, so the solver's own tolerance cannot hide one. A small reported value therefore certifies optimality to that size, and a real violation cannot be masked: no choice of multipliers brings it below the optimum of the program. With $r = G\beta - c + A^\top\kappa + C^\top\eta$ the conditions are $r_j = -\lambda \operatorname{sign} \beta_j$ on the interior of the support; $|r_j| \le \lambda$ at zero, or only $r_j \ge -\lambda$ under $\beta_j \ge 0$ and $r_j \le \lambda$ under $\beta_j \le 0$; $r_j \le -\lambda$ at a cap $u_j > 0$ and $r_j \ge \lambda$ at a floor $l_j < 0$; $\eta \ge 0$; and primal feasibility. The check covers each segment whole, not only its midpoint, as Hobson et al. (2026) do. Which coefficients are zero, at a bound or free, and which rows are active, is read off the midpoint. With that fixed, and the multipliers taken affine between the two ends, every condition is affine in λ. It holds along the whole segment if it holds at both ends. A linear program finds multipliers at both ends under the midpoint's pattern, and both ends must fit that pattern: inside a segment no coefficient may change sign or leave a bound, and no row may come loose. A path that is right at both ends of a segment and at its midpoint, but cuts across a knot, fails this check. The published tests include one (`test_segment_check_sees_between_samples`). With $p = 1$, $y = 1$ and the cap $\beta \le \frac{1}{2}$, the path is zero down to $\lambda = 1$, then $1 - \lambda$ up to the cap at $\lambda = \frac{1}{2}$. A single segment from $(\lambda, \beta) = (\frac{5}{4}, 0)$ to $(\frac{1}{4}, \frac{1}{2})$ misses both knots, yet meets the path at its ends and at its midpoint, where the largest violation of the optimality conditions is 0. The segment check returns 0.25, relative to $\|c\|_\infty$. The largest violation, relative to $\|c\|_\infty$, is the last column of the table.

Table 2 The CLA against LARS (knot by knot) and against an interior-point solver (IPM, at segment midpoints). *Gap*: largest discrepancy, relative to the largest coefficient. *KKT*: largest violation of the optimality conditions of (1) at every knot and along every segment, relative to $\|c\|_\infty$.

problem	constraint	n	p	knots	ref.	gap	KKT
plain	none	442	10	13	LARS	6.9×10^{-14}	1.1×10^{-15}
nonnegative	$\beta \geq 0$	442	10	6	LARS	2.5×10^{-15}	1.1×10^{-15}
wide	none, $p > n$	40	100	53	LARS	4.2×10^{-13}	6.1×10^{-16}
box	$\lvert\beta_j\rvert \leq u$	442	10	13	IPM	4.8×10^{-8}	1.2×10^{-15}
cholesterol	$\lvert\beta_{s1} + \beta_{s2} + \beta_{s3}\rvert \leq h$	442	10	14	IPM	3.4×10^{-9}	1.1×10^{-15}
pinned	$x_1^\top \beta = y_1$	442	10	18	IPM	3.6×10^{-10}	1.6×10^{-15}
zero-sum	$\mathbf{1}^\top \beta = 0$	60	30	30	IPM	1.1×10^{-10}	1.0×10^{-15}
zero-sum, signs	$\mathbf{1}^\top \beta = 0, \beta_{2j} \geq 0$	60	30	21	IPM	2.1×10^{-11}	1.0×10^{-15}
zero-sum, caps	$\mathbf{1}^\top \beta = 0, \lvert\beta_j\rvert \leq u$	60	30	35	IPM	1.4×10^{-9}	5.3×10^{-16}
monotone	$\beta_j \leq \beta_{j+1}$	100	10	13	IPM	4.6×10^{-12}	9.5×10^{-16}
sum to one	$\mathbf{1}^\top \beta = 1$	40	20	19	IPM	1.3×10^{-9}	7.5×10^{-16}
sCD14	$\mathbf{1}^\top \beta = 0$	151	60	60	IPM	8.8×10^{-10}	3.1×10^{-15}

5.2 *Accuracy*

The CLA returns LARS's knots to 10^{-13}, including on the wide design, where 53 knots carry the path to the interpolating fit. The KKT residuals are at the level of rounding throughout, including on the problems LARS cannot take. In positive mode `lars_path` labels as $\alpha = 0$ a point that is the path at $\lambda = 30.65$, so the CLA's last knot is compared with `scipy.optimize.nnls` (Lawson & Hanson, 1974) instead.

5.3 *Scale and cost*

Table 3 extends the test to simulated zero-sum problems of up to $p = 500$ parts, each size drawn with three seeds. At that size a midpoint solve per segment would cost more than the path itself, so the table reports only the optimality conditions. They hold to rounding. A second check at the largest size comes from the other encoding. The penalty form, relaxed, returns the same 542 knots to 8.5×10^{-15}, as it does on every zero-sum problem here (`figures/encodings.tex`). For a sense of cost, take a grid of 100 interior-point solves (Clarabel through CVXPY, compiled once with λ as a parameter). It costs about 90 times as much as the whole CLA path at $p = 100$, 340 times at $p = 500$, and 14 to 110 times on the constrained diabetes and zero-sum problems of Table 2. Each step was run once to warm up and then timed 11 times, 3 times at the sizes of Table 3, on an Apple M4 Pro. The ratios are of the medians; the ratio of any two paired runs stays within 8% of it. These are the only numbers the script does not write into the text. The grid also misses every knot between grid points.

Table 3 The budget form on simulated zero-sum problems of growing size, over three seeds: the range of the knot counts, and the largest KKT violation at every knot and along every segment, as in Table 2.

n	p	knots	KKT
200	100	100–108	2.1×10^{-15}
400	200	208–224	3.2×10^{-15}
1000	500	538–542	2.1×10^{-15}

5.4 *Hostile problems*

Table 4 adds problems chosen to make a path algorithm fail, each drawn with ten seeds (7 to 16), and reports the worst case. *Correlated* is a Gaussian design with $n = 100$ and $p = 40$ in which every pair of columns has correlation 0.99999. It is run alone, under a box and under a zero sum. *Duplicate* adds to the diabetes data an eleventh column, `bmi` plus noise of 1.0×10^{-4} or 1.0×10^{-6} times its spread, so that two columns nearly coincide; it is run alone and under the cap of *Box*. *Wide, zero-sum* puts a zero sum on *Wide*, where $p > n$.

Every path that is traced meets the optimality conditions to 9.6×10^{-12} or better. The gaps to the references grow with the conditioning. Against LARS the gap stays below cond $(X)^2$ times the machine epsilon, and LARS's own KKT residual is at most 2.0×10^{-13}: both paths are right. The same reading covers a minimiser that is not unique. LARS and the CLA may then trace different paths of minimisers, and a gap between them is no defect as long as both meet the optimality conditions; the certificate, not the agreement, is what each path is held to. Against the interior-point solver the gap reaches 1.5×10^{-2} on *Duplicate*, where two nearly identical columns leave the minimiser all but unique: the solver can split the weight between the twins differently at almost the same objective. Even there the solver's objective is never lower than the path's by more than 4.8×10^{-13}, relative, which is within its tolerance.

`cvxcla` declines *Duplicate* at 1.0×10^{-6} without a cap for every seed, because both twins become free (Section 6). Under the cap it declines two of the ten seeds, for the same reason: on those draws both twins become free before the cap binds. On the others they never are both free, and the path is exact. Nothing else is declined. *Wide, zero-sum*, with cond(X) near 10^{16}, is traced, with the largest residual in the table (Section 4).

Table 4 Hostile problems, through the checks of Table 2, the worst over ten seeds. *Declined*: `cvxcla` stops with an error. The duplicate column is `bmi` plus noise of the given size, relative to its spread.

problem	n	p	cond(X)	knots	ref.	gap	KKT
correlated	100	40	5.6×10^3	43–49	LARS	1.8×10^{-9}	3.9×10^{-13}
correlated, box	100	40	5.6×10^3	47–59	IPM	1.7×10^{-6}	3.0×10^{-13}
correlated, zero-sum	100	40	5.6×10^3	40–42	IPM	4.3×10^{-8}	5.1×10^{-13}
duplicate 10^{-4}	442	11	3.2×10^4	14–16	LARS	2.4×10^{-7}	3.2×10^{-13}
duplicate 10^{-4}, box	442	11	3.2×10^4	14–16	IPM	3.9×10^{-3}	4.0×10^{-14}
duplicate 10^{-6}	442	11	3.2×10^6	declined			
duplicate 10^{-6}, box	442	11	3.2×10^6	14 (2 declined)	IPM	1.5×10^{-2}	1.6×10^{-15}
wide, zero-sum	40	100	5.8×10^{15}	50–78	IPM	2.7×10^{-8}	9.6×10^{-12}

5.5 *More predictors than observations*

A zero sum alone with $p > n$ is covered by comlasso (Jeon et al., 2020) too, so Table 5 puts on *Wide* ($n = 40$, $p = 100$, ten seeds) the constraints comlasso does not take. They are nonnegativity; a box at half the largest coefficient of the least-squares fit of smallest norm; two binding inequality rows that cap the net effect of three predictors, as in *Cholesterol*; the row of *Pinned*; and a zero sum with the signs of *Zero-sum, signs* or with the caps of the box. `cvxcla` declines none of them. Every path meets the optimality conditions to 1.3×10^{-9} or better along every segment. The largest residuals come at the end of a path, near $\lambda = 10^{-4}\lambda_{\max}$, where the fit almost interpolates; elsewhere they are at rounding. The largest is on one seed of *Zero-sum, signs*, with cond(X) near 10^{16}. and the interior-point solver agrees to 4.2×10^{-6}. Its objective is never lower than the path's by more than 8.1×10^{-12}, relative. That happens only near the end of a path, where the fit almost interpolates and the objective is small.

Table 5 More predictors than observations, $n = 40$ and $p = 100$, under constraints beyond a zero sum, through the checks of Table 2 against the interior-point solver, the worst over ten seeds. None is declined.

Wide with	knots	gap	KKT
nonnegative	48–100	3.5×10^{-8}	5.6×10^{-15}
box	92–315	2.7×10^{-6}	1.3×10^{-14}
inequality rows	50–78	8.0×10^{-8}	1.4×10^{-11}
pinned	51–73	6.6×10^{-8}	9.7×10^{-15}
zero-sum, signs	44–78	9.3×10^{-8}	1.3×10^{-9}
zero-sum, caps	88–389	4.2×10^{-6}	2.2×10^{-12}

5.6 *Three independent path algorithms*

Three path algorithms derived for the LASSO, with no reference to portfolios, can be run against the CLA. Started at the exact first knot, all three trace its path, up to the ridge two of them add.

comlasso (Jeon et al., 2020) traces the exact path under zero sums on blocks of coefficients, with no ridge and with $p > n$, so it should agree with the CLA with no workaround. Its authors publish it as an R package built for R 3.2. We load it unchanged (Code availability) and run it on *Zero-sum*, *sCD14* and *Wide, zero-sum*. It returns the CLA's knots, to 4.3×10^{-15} relative to the largest coefficient on the first two and to 1.5×10^{-10} on *Wide, zero-sum* over the ten seeds.

The algorithm of Gaines et al. (2018) ships in MATLAB and in Julia. We ported the Julia package to Julia 1 without changing the algorithm (Code availability) and ran it on every problem of Table 2. Started at the exact first knot, where that is simple to state, it agrees with the CLA to 2.9×10^{-14} on the five such problems with $n \geq p$: box, cholesterol, plain, sCD14 and zero-sum. On *Wide*, where $n < p$, its ridge keeps it 2.8×10^{-2} away. As shipped, its own start puts it off the path on all 12 problems, by 1.6×10^{-5} to 1.1×10^{2} relative to the largest coefficient. It takes the first knot from a linear program that does not involve y, and calls a coefficient active above an absolute threshold of 10^{-4}.

`c-lasso` (Simpson et al., 2021) has a path algorithm for $C\beta = 0$ alone, without bounds or signs, so it runs on *Zero-sum*, *sCD14* and *Wide, zero-sum*. It adds a ridge term $10^{-3}I$ to $X^{\top}X$, fixed in the package, and that term is the whole of its distance to the CLA. On the first two its 100 grid points lie up to 6.1×10^{-5} from the CLA path as shipped, relative to the largest coefficient. With the ridge set to 10^{-12} by hand they lie 6.0×10^{-14} from it. With $p > n$ the same ridge is not small. On *Wide, zero-sum*, `c-lasso` as shipped lies 0.41 to 0.60 times the largest coefficient from the CLA over the ten seeds, and 1.5×10^{-10} with the ridge at 10^{-12}.

6 Limits and when to use it

When two columns nearly coincide and both become free, moving weight from one to the other barely changes the fit. Unless an active row forbids that move, (3) is then singular to working precision and `cvxcla` stops (*Duplicate* at 1.0×10^{-6}, Table 4). This is the general-position assumption of Section 1, failing in floating point. Dropping or merging one of the two columns before tracing avoids it. A relaxation can also fail its certificate, when knots of the exact path lie closer together than the relaxation moves them; here none was declined. Finally, we do not prove that the walk terminates. The non-terminating examples of Hobson et al. (2026) need multipliers that are not unique, which a nonsingular (3) rules out, and every walk here ended.

When to use it `cvxcla.Lasso` is for LASSO paths under constraints `lars_path` does not take: caps, signs on some coefficients, equality and inequality rows and zero sums, also with $p > n$. It returns the knots at less cost than a grid of interior-point solves (Section 5.3). For the plain or positive LASSO, `lars_path` is enough. Two nearly identical columns should be merged first. Table 6 lists what each path algorithm accepts.

https://doi.org/10.68365/jrepstat.2026.10 · CC-BY 4.0 · © 2026 The Author(s)

Table 6 What each exact path algorithm accepts. *Rows*: a bound or sign is written as an inequality row. *All*: `positive=True` makes every coefficient nonnegative, none can be chosen alone. `c-lasso` takes $C\beta = 0$; comlasso takes zero sums on blocks of coefficients. The ridges are 10^{-3} in `c-lasso`, 10^{-4} in Gaines et al. when $n < p$, and $10^{-4} \max_j G_{jj}$ in CLASSO, whose code was not released when we wrote; its column is from its paper (Hobson et al., 2026).

	`lars_path`	`c-lasso`	comlasso	Gaines et al.	CLASSO	`cvxcla`
caps $l \le \beta \le u$	–	–	–	rows	rows	yes
per-coefficient signs	all	–	–	rows	rows	yes
equality rows $A\beta = b$	–	$b = 0$	zero sums	yes	yes	yes
inequality rows $C\beta \le h$	–	–	–	yes	yes	yes
zero sums	–	yes	on blocks	yes	yes	yes
$p > n$	yes	yes	yes	yes	yes	yes
$p > n$ without a ridge	yes	–	yes	–	–	yes

We do not compare run times with comlasso, the algorithm of Gaines et al. or `c-lasso`. None runs here as its authors would run it today: comlasso is R code built for R 3.2 and read under R 4, the algorithm of Gaines et al. is our port of Julia 0.6 code, and `c-lasso` returns a grid of its own with a ridge. A timing across R, Julia and Python would measure the languages and the ports as much as the algorithms. A user choosing among them should first ask which constraints each accepts, and then time them on their own problem.

7 Conclusion

The checks here support the identity of Schmelzer and Hastie (2026) wherever the encodings reach. On every problem, a CLA written for portfolios returns the exact knots, or stops with an error where its arithmetic cannot be trusted. With $p > n$ the paths stay exact with no ridge, where the other path algorithms for general constraints trace a ridged problem that ends elsewhere.

Data and Code Availability

The software is in `preprints/constrained_lasso/` of https://github.com/Jebel-Quant/lasso, under the MIT licence. Each script has a lockfile that fixes Python 3.12 and every package by version and hash. `python/make_figures.py` computes every path with `cvxcla.Lasso` 2.7.1 (Schmelzer, 2026a), runs every check, and writes every figure, table and number the text quotes. `python/classo_check.py` runs `c-lasso` 1.0.11. `R/comlasso_check.R` runs comlasso 0.2 under R 4.6.1, locally and in CI. The package was built for R 3.2, and R 4 refuses to install a package built before R 4.0, so the script does not install it. It fetches the package from its authors' repository at a pinned commit, checks its SHA-256, and reads its functions from the package's serialized code with `lazyLoad`, unchanged. comlasso is plain R, with no compiled code, so nothing in it depends on the R 3 build; its one dependency, `Rglpk`, comes from a dated CRAN snapshot. `julia/classopath.jl` ports `ConstrainedLasso.jl`, written for Julia 0.6, to Julia 1.12, with each departure marked; none changes the algorithm. `julia/gaines_check.jl` runs it in a pinned environment. The sCD14 data are not redistributed: `python/fetch_scd14.py` downloads `coda4microbiome` 0.2.4 from CRAN or, should CRAN drop it, from a dated snapshot of Posit Package Manager. It checks the SHA-256 of the tarball, and that of the data file inside it, which is the same byte for byte in both. `make figures` runs all five, in CI as well.

Every random design is drawn inside `python/make_figures.py` from a seed recorded there: seed 7 for the problems of Table 2, seeds 7 to 9 for Table 3, and seeds 7 to 16 for Tables 4 and 5. On a fixed platform the discrete results reproduce exactly: the knot counts, which problems are declined, and whether each path passes the certificate. A test runs the problems of Tables 3 and 4 twice and checks that they do. Only residuals near rounding vary, and only in their last digits.

The lockfiles fix the software, not the hardware: those residuals also differ in the last digits on another CPU, so the tests use tolerances between 10^{-12} and 10^{-10}, and costs are given only as ratios.

Competing interests The author works at Jebel Quant Research, which hosts the code, and maintains `cvxcla`, which computes every path. The checks are the author's; the references they compare against are not. LARS, Clarabel, comlasso, `ConstrainedLasso.jl` and `c-lasso` were written by others, and the optimality conditions need no solver. There are no other competing interests.

Acknowledgements I thank Trevor Hastie for many helpful discussions.

References

Abadie, A., Diamond, A., & Hainmueller, J. (2010). Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California's Tobacco Control Program. *Journal of the American Statistical Association*, *105*(490), 493–505. https://doi.org/10.1198/jasa.2009.ap08746

Brodie, J., Daubechies, I., De Mol, C., Giannone, D., & Loris, I. (2009). Sparse and stable Markowitz portfolios. *Proceedings of the National Academy of Sciences*, *106*(30), 12267–12272. https://doi.org/10.1073/pnas.0904287106

Calle, M. L., Pujolassos, M., & Susín, A. (2023). coda4microbiome: compositional data analysis for microbiome cross-sectional and longitudinal studies. *BMC Bioinformatics*, *24*, 82. https://doi.org/10.1186/s12859-023-05205-3

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, *32*(2), 407–499. https://doi.org/10.1214/009053604000000067

Gaines, B. R., Kim, J., & Zhou, H. (2018). Algorithms for Fitting the Constrained Lasso. *Journal of Computational and Graphical Statistics*, *27*(4), 861–871. https://doi.org/10.1080/10618600.2018.1473777

Hobson, M., Butcher, L., & Lentati, L. (2026). *CLASSO: a practical primal path-following algorithm for the constrained LASSO*. arXiv. https://doi.org/10.48550/ARXIV.2610.17024

James, G. M., Paulson, C., & Rusmevichientong, P. (2020). Penalized and Constrained Optimization: An Application to High-Dimensional Website Advertising. *Journal of the American Statistical Association*, *115*(529), 107–122. https://doi.org/10.1080/01621459.2019.1609970

Jeon, J.-J., Kim, Y., Won, S., & Choi, H. (2020). Primal path algorithm for compositional data analysis. *Computational Statistics & Data Analysis*, *148*, 106958. https://doi.org/10.1016/j.csda.2020.106958

Lawson, C. L., & Hanson, R. J. (1974). *Solving Least Squares Problems*. Prentice-Hall.

Lin, W., Shi, P., Feng, R., & Li, H. (2014). Variable selection in regression with compositional covariates. *Biometrika*, *101*(4), 785–797. https://doi.org/10.1093/biomet/asu031

Markowitz, H. M. (1956). The optimization of a quadratic function subject to linear constraints. *Naval Research Logistics Quarterly*, *3*(1–2), 111–133. https://doi.org/10.1002/nav.3800030110

Noguera-Julian, M., Rocafort, M., Guillén, Y., Rivera, J., Casadellà, M., Nowak, P., Hildebrand, F., Zeller, G., Parera, M., Bellido, R., Rodríguez, C., Carrillo, J., Mothe, B., Coll, J., Bravo, I., Estany, C., Herrero, C., Saz, J., Sirera, G., … Paredes, R. (2016). Gut Microbiota Linked to Sexual Preference and HIV Infection. *EBioMedicine*, *5*, 135–146. https://doi.org/10.1016/j.ebiom.2016.01.032

Okabe, M., & Ito, K. (2008). *Color universal design (CUD): how to make figures and presentations that are friendly to colorblind people*. https://jfly.uni-koeln.de/color/

Osborne, M. R., Presnell, B., & Turlach, B. A. (2000). A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, *20*(3), 389–403. https://doi.org/10.1093/imanum/20.3.389

Pedregosa, F. et al. (2011). Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Schmelzer, T. (2026b). *cvxcla: The Critical Line Algorithm with an Operator Interface in Python*. Elsevier BV. https://doi.org/10.2139/ssrn.7583179

Schmelzer, T. (2026a). The efficient frontier from a LASSO solver: a numerical check of the CLA–LASSO identity. *Journal of Reproducible Statistics*, *1*. https://doi.org/10.68365/jrepstat.2026.7

Schmelzer, T., & Hastie, T. (2026). *The Critical Line Algorithm and the Constrained LASSO: One Curve, Two Literatures*. arXiv. https://doi.org/10.48550/ARXIV.2609.25704

Simpson, L., Combettes, P. L., & Müller, C. L. (2021). c-lasso – a Python package for constrained sparse and robust regression and classification. *Journal of Open Source Software*, *6*(57), 2844. https://doi.org/10.21105/joss.02844

Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Tibshirani, R. J. (2013). The lasso problem and uniqueness. *Electronic Journal of Statistics*, *7*, 1456–1490. https://doi.org/10.1214/13-EJS815

Wilson, P. W. F., Meigs, J. B., Sullivan, L., et al. (2007). Prediction of incident diabetes mellitus in middle-aged adults: the Framingham Offspring Study. *Archives of Internal Medicine*, *167*(10), 1068–1074. https://doi.org/10.1001/archinte.167.10.1068

COLOPHON Typeset with Typst 0.15.0 from the author's LaTeX source. Template jrepstat.typ at commit 004e37d. Set in STIX Two Text, STIX Two Math and JetBrains Mono. Built 1 November 2026.