Linear least-squares over the unit simplex

Working note RAL-NA-2023–2 — Nicholas I. M. Gould

14th March 2023

1 Introduction

We are concerned with the simplex-constrained linear least-squares problem

$$\underset{x\in\Delta^n}{\text{minimize}}\ f(x) := \frac{1}{2}\|Ax-b\|^2, \tag{1.1}$$

where the m by n matrix A and m vector b are given, the unit simplex

$$\Delta^n := \{x \mid \sum_{i=1}^{n} x_i = 1,\ x_i \geq 0,\ i = 1,\ldots,n\},$$

and $\|\cdot\|$ denotes the Euclidean (ℓ_2) norm; extension to the weighted case in which the objective function is instead $\frac{1}{2}\|Ax-b\|^2 = \frac{1}{2}(Ax-b)^T W(Ax-b)$ for some positive definite (diagonal) matrix W is obvious, and although we give no details here, our software copes with this possibility. Let $P(v)$ be the projection of a given v into the simplex Δ^n; we will return shortly to see how this may be calculated. In addition, let

$$\mathcal{A}(v) := \{1 \leq i \leq n \mid P(v)_i = 0\} \tag{1.2}$$

be the set of indices of variables that are zero at $P(v)$. We let e be the vector of ones of appropriate length, and $s_{\mathcal{I}}$ is the vector whose components are the components $[s]_i$ of a given vector s for $i \in \mathcal{I}$, where $\mathcal{I} \subseteq \mathcal{N} := \{1,\ldots,n\}$. We will abuse notation slightly in what follows: a subscript k will refer to an iterate (and may be a vector or matrix), while other subscripts i, j, etc., will be components of vectors and higher-order tensors (if needed the component of a vector iterate will have a subscript i,k).

To put things into perspective, problem (1.1) is a special but important instance of the more general single-constrained quadratic program

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \frac{1}{2}x^T Hx + c^T x \ \text{ subject to }\ a^T x = \beta \ \text{ and }\ x^{\rm L} \leq x \leq x^{\rm U}, \tag{1.3}$$

for a given symmetric positive-semi-definite matrix H, vectors $c, a, x^{\rm L}$ and $x^{\rm U}$ and scalar β. Methods of various types have been proposed for this, those based on active-sets, interior-points, and gradient projection [8, 9] being the most common. The case where the feasible

set in (1.3) is the simplex Δ^n is commonly called a standard quadratic program, although much of the focus in that case is devoted to the nonconvex case for which H may have negative eginvalues [1].

The specific problem (1.1) occurs naturally in multi-spectral un-mixing [2, 5, 17], while we were drawn to it as a subproblem for alternating methods [3] that aim to fit a multilinear form $T[x_i, \ldots, x_m]$ to data, in a weighted least-squares sense, from polydisperse small-angle scattering applications, e.g., [19]. Here T is a given m-way tensor, the vectors of unknowns x_i inherit dimensions from T, and the alternating methods successively fix $m-1$ of them and find the remainder by least-sqaures fitting subject to a normalizing simplex constraint.

2 An algorithm

2.1 A basic algorithm

A basic method to find an approximate solution to (1.1) is as follows:

Algorithm 2.1: linear least-squares over the unit simplex

Step 0: Initialization. Given a convergence tolerance $\epsilon > 0$, pick an initial estimate $x_0 \in \Delta^n$, and set a counter $k = 0$.

Step 1: Test for termination. Compute the residual $r_k = Ax_k - b$ and gradient $g_k = A^T r_k$. If

$$\|P(x_k - g_k) - x_k\| \leq \epsilon,$$

stop with the approximate solution $x_* := x_k$.

Step 2: Compute a Cauchy point. Compute the Cauchy direction

$$s_k^{\rm C} = \ \arg\min_{s \in \mathbb{R}^n} \ g_k^T s \ \text{ subject to } \ e^T s = 0 \ \text{ and } \ s \geq -x_k, \tag{2.1}$$

the stepsize $\alpha_k^{\rm C} > 0$ that sufficiently reduces $f(P(x_k + \alpha s_k^{\rm C}))$, and the Cauchy point $x_k^{\rm C} = P(x_k + \alpha_k^{\rm C} s_k^{\rm C})$.

Step 3: Improve on the Cauchy point. If desired, compute an approximate solution s_k to the *equality-constrained* linear least-squares problem

$$\underset{s \in \mathbb{R}^n}{\rm minimize} \ \frac{1}{2}\|A(x_k^{\rm C} + s) - b\|^2 \text{ subject to } e^T s = 0 \text{ and } s_{\mathcal{A}(x_k^{\rm C})} = 0. \tag{2.2}$$

Otherwise define the next iterate $x_{k+1} = x_k^{\rm C}$ and go to Step 5.

Step 4: Find the next iterate. Find a stepsize $\alpha_k > 0$ that reduces $f(P(x_k^{\mathrm{C}}+\alpha s_k))$, and define the next iterate $x_{k+1} = P(x_k + \alpha_k s_k)$.

Step 5: Perform the next iteration. Increase k by 1 and go to Step 1.

A few comments are in order. Firstly, in Steps 2 and 4 we need to consider the least-squares objective function $f(x)$ along a piecewise arc $x(\alpha; x, v) = P(x + \alpha v)$ emanating from a feasible point x in the direction v but bent so as to remain feasible with respect to Δ^n. We consider this further in due course. We have not specified what "approximate" means here, but two possibilities are either to find the exact minimizer along the arc, or to find a point that gives "sufficient" decrease to guarantee convergence.

Secondly, it is trivial to find the required Cauchy direction in (2.1). For the problem in question is clearly a linear program, and as the feasible set is bounded, its solution must occur at a vertex, that is a point at which $n-1$ of the bounds $s \geq x_k$ are satisfied as equalities. Suppose that variable i is the exception. Then

$$s_j = -x_{j,k} \text{ for } j \neq i \text{ and } s_i = 1 - x_{i,k} \tag{2.3}$$

since $x_k \in \Delta^n$ and $e^T s = 0$. Hence, for this s, the objective

$$g_k^T s = \sum_{j=1}^{n} g_{j,k} s_j = -\sum_{j=1}^{n} g_{j,k} x_{k,j} + g_{i,k}$$

using (2.3). The smallest value of $g_k^T s$ over all such vertices then occurs for the index (perhaps indices) for which $g_{i,k}$ is smallest, and we use such an i with (2.3) to define s_k^{C}.

Thirdly, notice that Steps 3 and 4 are optional; the algorithm is guaranteed to converge without them, but almost always moving beyond the Cauchy point improves performance. From a theoretical perspective, the Cauchy point plays two roles, it guarantees convergence and in non-degenerate cases it also ultimately predicts the optimal face on the unit simplex. Once this happens, the exact solution $x_k^{\mathrm{C}} + s_k$ from (2.2) will also be that of (1.1), and $\alpha_k = 1$ will result from Step 4. Steps 3 and 4 might be skipped if, for example, the set $\mathcal{A}(x_k^{\mathrm{C}})$ differs significantly from its immediate predecessor.

Finally, the equality-constrained linear least-squares problem (2.2) may be written more compactly as

$$\underset{s_{\mathcal{I}_k}}{\text{minimize}} \ \frac{1}{2}\|A_{\mathcal{I}_k} s_{\mathcal{I}_k} - (b - Ax_k^{\mathrm{C}})\|^2 \text{ subject to } e^T s_{\mathcal{I}_k} = 0 \tag{2.4}$$

to determine the "free" components $s_{\mathcal{I}_k}$ of s, where $\mathcal{I}_k = \mathcal{N} \setminus \mathcal{A}(x_k^{\mathrm{C}})$. That is, it only involves the "free" columns $A_{\mathcal{I}_k}$ of A. To solve a generic singly-constrained linear least-squares problem

$$\underset{s \in \mathbb{R}^n}{\text{minimize}} \ m(s) := \frac{1}{2}\|As - b\|^2 \text{ subject to } e^T s = 0 \tag{2.5}$$

at least two possibilities are available.

The first is to note that the solution to (2.5) must satisfy the optimality conditions

$$A^T(As-b)+\lambda e=0 \text{ and } e^Ts=0$$

for some scalar Lagrange multiplier λ. This then leads to the linear system

$$\begin{pmatrix} A^TA & e \\ e^T & 0 \end{pmatrix}\begin{pmatrix} s \\ \lambda \end{pmatrix}=\begin{pmatrix} A^Tb \\ 0 \end{pmatrix}$$

or, on defining $r=b-As$, the expanded system

$$\begin{pmatrix} I & A & 0 \\ A^T & 0 & e \\ 0 & e^T & 0 \end{pmatrix}\begin{pmatrix} r \\ s \\ \lambda \end{pmatrix}=\begin{pmatrix} b \\ 0 \\ 0 \end{pmatrix}.$$

or even its solution via

$$\begin{pmatrix} r \\ s \end{pmatrix}=\begin{pmatrix} p \\ q \end{pmatrix}-\lambda\begin{pmatrix} u \\ v \end{pmatrix}, \text{ where } \begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix}\begin{pmatrix} p & u \\ q & v \end{pmatrix}=\begin{pmatrix} b & 0 \\ 0 & e \end{pmatrix} \text{ and } \lambda=\frac{q^Te}{v^Te}.$$

Since we are primarily only interested in the s component of the solution, the latter may be rewritten as

$$s=q-\lambda v, \text{ where } A^TAq=A^Tb,\ A^TAv=-e \text{ and } \lambda=\frac{q^Te}{v^Te}. \tag{2.6}$$

Each of the above systems may be solved using symmetric, indefinite factorization, and in the special case (2.6), Cholesky factorization of A^TA suffices.

A second possibility is to solve—possibly approximately—the problem using an iterative method. Standard techniques such as the preconditioned conjugate-gradient normal equation (CGNE) method may be applied so long as the preconditioning step respects the single linear constraint. That is, given a gradient g_k of $m(s)$ at CGNE iterate s_k, the required slope $g_k^Tv_k$ is computed to satisfy

$$\begin{pmatrix} P & e \\ e^T & 0 \end{pmatrix}\begin{pmatrix} v_k \\ \mu_k \end{pmatrix}=\begin{pmatrix} g_k \\ 0 \end{pmatrix} \tag{2.7}$$

for some suitable symmetric approximation P to A^TA. This is a special case of the projected conjugate gradient method [13] applied to (2.5), and here as in [13, §5] it is very important to perform at least one iterative refinement per solve (2.7) to ensure that the projected conjugate gradient iterates do not deviate significantly from the manifold $e^Ts_{\mathcal{I}_k}=0$. Of course the solution to (2.7) may be expressed formally as

$$v_k=P^{-1}g_k-\mu_kP^{-1}e, \text{ where } \mu_k=\frac{e^TP^{-1}g_k}{e^Te}$$

which is useful if P has a simple form, e.g., if P is diagonal with $P=\text{diag}(A^TA)$ being a common, often effective [15] option. Constrained variants of other specialised iterative least-squares methods such as LSQR [21], LSMR [11] or LSLQ [10] may be preferred.

2.2 Projections onto the unit simplex and projection arcs

2.2.1 Projections onto the unit simplex

Given the vector v, we let $v_{(i)}$, $i = 1, \ldots, n$, be the value of its i-th largest component. The basic $O(n \log n)$ method for finding the projection $P(v)$ is due to [16], and has been improved over the years, see [4, 7, 23] and the citations within. A typically efficient method is as follows:

Algorithm 2.2: projection onto the unit simplex via heapsort [23]

1. Build a heap $\mathcal{H}$ with largest entry h from the components of v.
2. For j = 1,. . . ,n, do
   - 2a. Set $v_{(j)} := h$.
   - 2b. If
   $$\left(\sum_{i=1}^{j} v_{(i)} - 1 \right) / j \geq v_{(j)},$$
   exit the loop.
   - 2c. Set $j_{\max} := j$.
   - 2d. Remove h from $\mathcal{H}$, and restore the heap $\mathcal{H}$ with new largest entry h
3. Define
   $$\tau := \left(\sum_{j=1}^{j_{\max}} v_{(j)} - 1 \right) / j_{\max}.$$
4. Set $P(v)_i := \max[v_i - \tau, 0]$ for $i = 1, \ldots, n$.

Such a method is based on satisfying the optimality conditions for the projection problem

$$P(v) = \arg\min_{p \in \Delta^n} \frac{1}{2} \|p - v\|^2$$

by the constructive finite iteration in Step 2.

2.2.2 Projection arcs

We next consider the arc $P(x + \alpha d)$ from a given $x \in \Delta^n$ in the generic direction d for positive α. Since x lies within Δ^n, the arc will be piecewise linear as α increases, and will change direction whenever one (or more) of its components shrinks to zero (and is fixed

thereafter). We refer to the points at which the direction changes as *breakpoints*, and the vectors that join adjacent breakpoints as *segments*.

Suppose that v is such a breakpoint, and that $\mathcal{A} := \mathcal{A}(v)$ is as in (1.2). Then the next segment s is in the direction

$$\arg\min_{s\in\mathbb{R}^n} \frac{1}{2}\|s-d\|^2 \text{ subject to } e^Ts = 0 \text{ and } s_{\mathcal{A}} = 0,$$

for which the necessary optimality conditions

$$s_{\mathcal{I}} - d_{\mathcal{I}} + \lambda e_{\mathcal{I}} = 0,\ e_{\mathcal{I}}^T s_{\mathcal{I}} = 0, \text{ and } s_{\mathcal{A}} = 0$$

give that s has components

$$s_{\mathcal{I}} = d_{\mathcal{I}} - \frac{e_{\mathcal{I}}^T d_{\mathcal{I}}}{e_{\mathcal{I}}^T e_{\mathcal{I}}} e_{\mathcal{I}} \text{ and } s_{\mathcal{A}} = 0 \tag{2.8}$$

where $\mathcal{I} := \mathcal{N} \setminus \mathcal{A}$.

Now suppose that at the next breakpoint v^+ variable i reaches zero, and thus that $\mathcal{I}^+ = \mathcal{I} \setminus \{i\}$. It then follows from (2.8) that the new segment s^+ emanating from x^+ satisfies

$$s^+_{\mathcal{I}^+} = d_{\mathcal{I}^+} - \frac{e_{\mathcal{I}^+}^T d_{\mathcal{I}^+}}{e_{\mathcal{I}^+}^T e_{\mathcal{I}^+}} e_{\mathcal{I}^+} \text{ and } s_{\mathcal{A}^+} = 0. \tag{2.9}$$

In particular, if $j \in \mathcal{I}^+ \subset \mathcal{I}$, it follows from (2.8) and (2.9) that

$$s_j = d_j - \frac{e_{\mathcal{I}}^T d_{\mathcal{I}}}{e_{\mathcal{I}}^T e_{\mathcal{I}}} \text{ and } s_j^+ = d_j - \frac{e_{\mathcal{I}^+}^T d_{\mathcal{I}^+}}{e_{\mathcal{I}^+}^T e_{\mathcal{I}^+}}$$

and particularly that

$$s_j^+ = s_j + \gamma_i, \text{ where } \gamma_i = \frac{e_{\mathcal{I}}^T d_{\mathcal{I}}}{e_{\mathcal{I}}^T e_{\mathcal{I}}} - \frac{e_{\mathcal{I}^+}^T d_{\mathcal{I}^+}}{e_{\mathcal{I}^+}^T e_{\mathcal{I}^+}} = \frac{e_{\mathcal{I}}^T d_{\mathcal{I}}}{|\mathcal{I}|} - \frac{e_{\mathcal{I}}^T d_{\mathcal{I}} - d_i}{|\mathcal{I}|-1} = \frac{|\mathcal{I}| d_i - e_{\mathcal{I}}^T d_{\mathcal{I}}}{|\mathcal{I}|(|\mathcal{I}|-1)} \tag{2.10}$$

since $e_{\mathcal{I}^+}^T e_{\mathcal{I}^+} = |\mathcal{I}^+| = |\mathcal{I}| - 1 = e_{\mathcal{I}}^T e_{\mathcal{I}} - 1$ and $e_{\mathcal{I}^+}^T d_{\mathcal{I}^+} = e_{\mathcal{I}}^T d_{\mathcal{I}} - d_i$. Thus each nonzero component of the new segment is that of the old segment shifted by *the same* γ_i. We may also use the relationships $e_{\mathcal{I}}^T s_{\mathcal{I}} = 0 = e_{\mathcal{I}^+}^T s_{\mathcal{I}^+}^+$ together with (2.10) to deduce that

$$0 = \sum_{j\in\mathcal{I}^+} s_j^+ = \sum_{j\in\mathcal{I}^+} (s_j + \gamma_i) = \sum_{j\in\mathcal{I}} s_j - s_i + \gamma_i|\mathcal{I}^+| = -s_i + \gamma_i|\mathcal{I}^+| \tag{2.11}$$

and thus we have the alternative definition

$$\gamma_i = \frac{s_i}{|\mathcal{I}^+|}. \tag{2.12}$$

Since $v_i + \alpha s_i = 0$, and $v_i > 0$, and $\alpha > 0$, we must have that $s_i < 0$ and consequently $\gamma_i < 0$. Hence the nonzero components of successive segments decrease; those that start negative stay so, while those that are initially positive move towards negativity.

We calculate breakpoints by examining the segment s emanating from v and picking

$$v^+ = v + ts, \quad \text{where } t_{\min} = \min_{s_j<0} t_j, \text{ where } t_j = -v_j/s_j. \tag{2.13}$$

By definition of i, we have $s_i < 0$ and $t_{\min} = t_i = -v_i/s_i$. Now suppose that $s_j < 0$ for component $j \neq i$. Then, as in the previous paragraph, $s_j^+ = s_j + \gamma_i$, and hence $s_j^+ < 0$. Moreover

$$t_j^+ - t_j = -\frac{v_j + ts_j}{s_j + \gamma_i} + \frac{v_j}{s_j} = \frac{v_j s_i^2/|\mathcal{I}^+| + v_i s_j^2}{s_j^+ s_i s_j} < 0$$

since the numerator is positive and the denominator negative, and hence $t_j^+ < t_j$. It might be tempting to hope that if $t_{\min} \leq t_j < t_k$ for some pair $j, k \neq i$ then $t_j^+ < t_k^+$. For if this were so, it would be possible to sort the indices at the beginning of the arc, and thereafter pick breakpoints sequentially according to their sorted order; this is one of the keys to efficiency in similar methods for the simpler box-shaped feasible domain $l \leq x \leq u$ [6, §17.3]. Unfortunately, this is not the case as the following example shows.

Consider the value $v = (0.01, 0.01, 0.05, 0.93) \in \Delta^4$, and the direction $s = d = (-10, -1, -10, 21)$ for which $e^T s = 0$. Then $t_{\min} = t_1 = 0.001 < t_3 = 0.005 < t_2 = 0.01$, and hence $t = 0.001$, giving $v^+ = (0, 0.009, 0.04, 0.951)$, $s^+ = (0.-4.3333, -13.3333, 17.6667)$ and $t_{\min}^+ = t_2^+ = 0.0020769 < t_3^+ = 0.003$. Thus it seems likely that it is necessary to sort indices at each breakpoint when deciding where the next segment ends—in practice, we have observed that it is common for the order to be retained, and so the lack of a guarantee is regrettable. This suggests that sorting algorithms for "mostly sorted" data, such as insertion sort [18] or introsort [20], might be profiably employed in many cases.

2.2.3 Objective function evolution along a projection arc

Our next goal is to consider the evolution of the objective function

$$f(P(x + \alpha d)) = \frac{1}{2}\|AP(x + \alpha d) - b\|^2$$

as α increases from 0 from $x \in \Delta^n$ in the segment direction d. As we have already seen, we may break the arc $P(x+td)$ into pieces, as we shall consider $v+ts$ from the breakpoint v along the segment s for $t \geq 0$. Thus, we are concerned with the convex quadratic

$$f(v + ts) = \frac{1}{2}\|r\|^2 + tf' + \frac{1}{2}t^2 f'', \text{ where } r := Av - b,\ p := As,\ f' := r^T p \text{ and } f'' := \|p\|^2.$$

There are clearly three possibilities. It might be that v is a (local) minimizer of f along the arc; this will happen if $f' \geq 0$. If this is not the case, either we may move to the end of the segment as $f(v + ts)$ continues to decrease, or we encounter a local minimizer en route. The latter occurs when

$$t_{\text{opt}} := -f'/f'' \leq t_{\min}.$$

Naively, one might simply evaluate these three possibilities by computing the product $p = As$ required by f' and f''. However, it is more efficient to recur the required quantities as the arc search proceeds.

Suppose that the search continues to the next breakpoint, v^+, and that variable i is reduced to zero, i.e., $\mathcal{I}^+ = \mathcal{I} \setminus \{i\}$. Recall from (2.8) that

$$p = As = A_{\mathcal{I}} s_{\mathcal{I}} = A_{\mathcal{I}^+} s_{\mathcal{I}^+} + a_i s_i,$$

where a_i is the i-th column of A, and from (2.10) that

$$p^+ = As^+ = A_{\mathcal{I}^+} s^+_{\mathcal{I}^+} = A_{\mathcal{I}^+}(s_{\mathcal{I}^+} + \gamma_i e_{\mathcal{I}^+}),$$

and hence that

$$p^+ = p + \gamma_i A_{\mathcal{I}^+} e_{\mathcal{I}^+} - s_i a_i.$$

But if $w := A_{\mathcal{I}} e_{\mathcal{I}}$, we may recur

$$w^+ = A_{\mathcal{I}^+} e_{\mathcal{I}^+} = A_{\mathcal{I}} e_{\mathcal{I}} - a_i = w - a_i \tag{2.14}$$

and subsequently

$$p^+ = p + \gamma_i w^+ - s_i a_i. \tag{2.15}$$

Crucially, both recurrences simply need access to a single—possibly sparse—column of A. In addition, we may recur

$$r^+ = Av^+ - b = A(v + t_{\min} s) - b = r + t_{\min} p \tag{2.16}$$

and calculate

$$\frac{1}{2}\|r^+\|^2 = \frac{1}{2}\|r\|^2 + t_{\min} f' + \frac{1}{2} t^2_{\min} f'', \; f'^+ = r^{+T} p^+ \text{ and } f''^+ = \|p^+\|^2. \tag{2.17}$$

The main work involved are the two potentially-sparse and two dense vector additions in (2.14)–(2.16), and two inner products in (2.17), and avoids a more costly matrix-vector product As. This is less efficient than projected search when the feasible region is a box [12], but this appears to be inevitable because of the complicating constraint $e^T x = 1$.

Another possibility is to aim for a value along the arc that provides sufficient decrease of $f(P(x + \alpha d))$ to ensure convergence rather than for the minimizer. Such a possibility has been examined in more general contexts using Goldstein-like conditions [6, §12.2], but we prefer a slightly simpler Armijo version such as that in [22]. Essentially, given an initial estimate α_0 of the optimal step length α, $f(P(x + \alpha d))$ is computed for a sequence of decreasing values $\alpha_k = \alpha_0 \beta^k$, $k \geq 0$ with $\beta \in (0, 1)$, and the search terminated at the first k for which

$$f(P(x + \alpha_k d)) \leq f(x) + \eta (\nabla_x f(x))^T (P(x + \alpha_k d) - x) \tag{2.18}$$

for given (typically small) $\eta \in (0, 1)$. If we write $s_k = P(x + \alpha_k d) - x$ and $r_k = AP(x + \alpha_k d) - b$, then $r_k = r + As_k$, where we recall that $r = Ax - b$. Hence we may rewrite (2.18) as

$$\frac{1}{2} r_k^T r_k \leq \frac{1}{2} r^T r + \eta (r^T r_k - r^T r).$$

Thus each k requires the computation of a projection $P(x+\alpha_k d)$, a matrix-vector product As_k and a pair of new inner products $r^T r_k$ and $r_k^T r_k$. How this compares to the earlier, exact method depends entirely on the choice of α_0, and a good choice isn't obvious. Indeed, a large initial α_0 may result in a wasteful sequence of identical evaluation points $P(x+\alpha_k d)$.

2.3 Regularized objective function evolution along a projection arc

A related problem is the simplex-constrained quadratically-regularized linear least-squares problem

$$\underset{x\in\Delta^n}{\text{minimize}}\ r(x) := f(x) + \frac{1}{2}\sigma\|x\|^2 \equiv \frac{1}{2}\|Ax-b\|^2 + \frac{1}{2}\sigma\|x\|^2, \tag{2.19}$$

for some weight $\sigma \geq 0$. Much of what we have said so far in Section 1 holds with minor modification—such as replacing g_k by $A^T r_k + \sigma x_k$, adding the term $\frac{1}{2}\sigma\|x_k^{\mathrm{C}} + s\|^2$ to the objective in (2.2), and adding/subtracting the term σI from appropriaate blocks of the systems on p.4—and the only detail we address is how the regularized objective function

$$r(P(x+\alpha d)) = \frac{1}{2}\|AP(x+\alpha d)-b\|^2 + \frac{1}{2}\sigma\|P(x+\alpha d)\|^2$$

evolves as α increases. We thus focus on the term $\|v+ts\|^2$ on the segment s emanating from the breakpoint v, at which components indexed by $\mathcal{I}$ are free, as $t \geq 0$ increases. Since

$$\|v+ts\|^2 = \rho + 2t\rho' + t^2\rho'', \ \text{ where } \ \rho := ||v||^2,\ \rho' := v^T s \text{ and } \rho'' := ||s||^2,$$

we need to examine how ρ and its derivatives behave along successive segments.

It follows from the discussion in Section 2.2.2 that if the segment finishes at the breakpoint v^+ at which the component $v_i^+ = 0$, and thus $\mathcal{I}^+ = \mathcal{I} \setminus \{i\}$, we have

$$s_j^+ = s_j + \gamma_i \ \text{ for } \ j \in \mathcal{I}^+ \ \text{ and } \ s_i^+ = 0$$

from (2.10), and

$$v_j^+ = v_j + t_i s_j \ \text{ for } \ j \in \mathcal{I}^+ \ \text{ and } \ v_i^+ = 0$$

from (2.13), where we recall that

$$0 = \sum_{j\in\mathcal{I}} s_j = \sum_{j\in\mathcal{I}^+} s_j + s_i \ \text{ and } \ 1 = \sum_{j\in\mathcal{I}} v_j = \sum_{j\in\mathcal{I}^+} v_j + v_i \tag{2.20}$$

because $v \in \Delta^n$ and

$$\gamma_i|\mathcal{I}^+| = s_i \ \text{ and } \ v_i + t_i s_i = 0 \tag{2.21}$$

which follow directly from (2.12) and (2.13). Thus

$$\begin{aligned}\rho''^+ &= \sum_{j\in\mathcal{I}^+}(s_j+\gamma_i)^2 = \sum_{j\in\mathcal{I}^+} s_j^2 + 2\gamma_i \sum_{j\in\mathcal{I}^+} s_j + \gamma_i^2|\mathcal{I}^+| = \rho'' - s_i^2 - 2\gamma_i s_i + \gamma_i^2|\mathcal{I}^+| \\ &= \rho'' - s_i^2 - \gamma_i s_i\end{aligned} \tag{2.22}$$

using (2.20) and (2.21). Likewise

$$\begin{aligned}\rho'^{+} &= \sum_{j\in\mathcal{I}^+}(s_j+\gamma_i)(v_j+t_is_j) = \sum_{j\in\mathcal{I}^+}s_jv_j + t_i\sum_{j\in\mathcal{I}^+}s_j^2 + \gamma_i\sum_{j\in\mathcal{I}^+}v_j + \gamma_it_i\sum_{j\in\mathcal{I}^+}s_j\\ &= \sum_{j\in\mathcal{I}+}s_jv_j - s_iv_i + t_i\left(\sum_{j\in\mathcal{I}}s_j^2 - s_i^2\right) + \gamma_i(1-v_i) - \gamma_it_is_i \qquad (2.23)\\ &= \rho' + t_i\rho'' - s_i(v_i+t_is_i) + \gamma_i - \gamma_i(v_i+t_is_i)\\ &= \rho' + t_i\rho'' + \gamma_i\end{aligned}$$

again using (2.20) and (2.21), and of course

$$\rho^+ = \|v+t_is\|^2 = \rho + 2t_i\rho' + t_i^2\rho''. \qquad (2.24)$$

Thus it is trivial to update ρ and its derivatives using (2.22)–(2.24) as the arc evolves.

Availability

The algorithms described have been implemented as the modern Fortran package `sls`, and the later is available as part of the GALAHAD library [14].

References

[1] I. M. Bomze. On standard quadratic optimization problems. *Journal of Global Optimization*, 13:369—-387, 1998.

[2] J. Chen, C. Richard, H. Lantéri, C. Theys, and P. Honeine. A gradient based method for fully constrained least-squares unmixing of hyperspectral images. In *Proc. IEEE workshop on Statistical Signal Processing (SSP)*, pages 301–304, Nice, France, 2011.

[3] P. Comon, X. Luciani, and A. L. F. de Almeida. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, 23(7-8):393–405, 2009.

[4] L. Condat. Fast projection onto the simplex and the ℓ_1 ball. *Mathematical Programming*, 158(1–2):575—-585, 2016.

[5] L. Condat. Least-squares on the simplex for multispectral unmixing. Research report, GIPSA-Lab, Grenoble, France, 2017.

[6] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. *Trust-Region Methods*. SIAM, Philadelphia, 2000.

[7] Y. Dai and C. Chen. Distributed projections onto a simplex. arXiV.2204.08153, 2022.

[8] Y.-H. Dai and R. Fletcher. New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Mathematical Programming*, 106(3):409–421, 2006.

[9] D. di Serafino, G. Toraldo, M. Viola, and J. L. Barlow. A two-phase gradient method for quadratic programming problems with a single linear constraint and bounds on the variables. *SIAM Journal on Optimization*, 20(4):2809–2838, 2018.

[10] R. Estrin, D. Orban, and M. A. Saunders. LSLQ: An iterative method for linear least squares with an error minimization propert. *SIAM Journal on Matrix Analysis and Applications*, 40(1):254–275, 2019.

[11] D. C.-L. Fong and M. A. Saunders. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971, 2011.

[12] N. I. M. Gould. A projection method for bound-constrained linear least-squares. Working Note RAL 2023-1, STFC-Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2023.

[13] N. I. M. Gould, M. E. Hribar, and J. Nocedal. On the solution of equality constrained quadratic problems arising in optimization. *SIAM Journal on Scientific Computing*, 23(4):1375–1394, 2001.

[14] N. I. M. Gould, D. Orban, and Ph. L. Toint. GALAHAD—a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. *ACM Transactions on Mathematical Software*, 29(4):353–372, 2003.

[15] N. I. M. Gould and J. A. Scott. The state-of-the-art of preconditioners for sparse linear least-squares problems. *ACM Transactions on Mathematical Software*, 43(4), 2017. Article 36.

[16] M. Held, P. Wolfe, and H. P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88, 1974.

[17] R. Heylen, D. Burazerovic, and P. Scheunders. Fully constrained least squares spectral unmixing by simplex projection. *IEEE Transactions on Geoscience and Remote Sensing*, 49(11):4112–4122, 2011.

[18] D. E. Knuth. *The Art of Computer Programming, Volume 3, Sorting and Searching*. Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1973.

[19] K. Leng, S. King, T. Snow, S. Rogers, A. Markvardsen, S. Maheswaran, and J. Thiyagalingam. Parameter inversion of a polydisperse system in small-angle scattering. *Journal of Applied Crystallography*, 55(4):966–977, 2022.

[20] D. R. Musser. Introspective sorting and selection algorithms. *Software: Practice and Experience*, 27(8):983–993, 1997.

[21] C. C. Paige and M. A. Saunders. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software*, 8(1):43–71, 1982.

[22] A. Sartenaer. Armijo-type condition for the determination of a generalized Cauchy point in trust region algorithms using exact or inexact projections on convex constraints. *Belgian Journal of Operations Research, Statistics and Computer Science*, 33(4):61–75, 1993.

[23] E. van den Berg and M. P. Friedlander. Probing the Pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890—-912, 2008.