A projection method for bound-constrained linear least-squares

Working note RAL-NA-2023–1 — Nicholas I. M. Gould

10th March 2023

1 Introduction

We consider the bound-constrained linear least-squares problem

$$\underset{x \in \mathcal{X}}{\text{minimize}} \;\; f(x) := \tfrac{1}{2} \|Ax - b\|^2 \;\; \text{where} \;\; \mathcal{X} := \{x \in \mathbb{R}^n : x^{\mathrm{L}} \leq x \leq x^{\mathrm{U}}\}. \tag{1.1}$$

Here, we are given a vector of m observations b, a linear model Ax that aims to match b, written in terms of an m by n matrix A, and vectors of lower and upper bounds $x^{\mathrm{L}}, x^{\mathrm{U}} \in \mathbb{R}^n$, some of whose components may be infinite. We are particularly concerned with problems for which n and m are large, and A is sparse.

Throughout we use the Euclidean inner product $\langle \cdot, \cdot \rangle$, the corresponding norm $\|\cdot\| = \|\cdot\|_2$, and the projection operator

$$P_{\mathcal{X}}[x] = \max(x^{\mathrm{L}}, \min(x^{\mathrm{U}}, x)),$$

where min and max are applied componentwise. We denote the residual by $r(x) := Ax - b$, and the gradient and (constant) Hessian of f by $g(x) := A^T r(x)$ and $H := A^T A$. We partition the indices of $x \in \mathcal{X}$ into bounded and free sets via

$$\mathcal{B}^{\mathrm{L}}(x) := \{j \in \mathbb{N}_n : [x]_j = [x^{\mathrm{L}}]_j\}, \;\; \mathcal{B}^{\mathrm{U}}(x) := \{j \in \mathbb{N}_n : [x]_j = [x^{\mathrm{U}}]_j\},$$

$$\mathcal{B}(x) := \mathcal{B}^{\mathrm{L}}(x) \cup \mathcal{B}^{\mathrm{U}}(x) \;\; \text{and} \;\; \mathcal{F}(x) := \{j \in \mathbb{N}_n : [x^{\mathrm{L}}]_j < [x]_j < [x^{\mathrm{U}}]_j\},$$

where $\mathbb{N}_n := \{1, \cdots, n\}$ and $[x]_j$ is the jth component of x. Finally we let $A_{\mathcal{J}}$ be the matrix whose columns are those of A indexed by the set $\mathcal{J} \subseteq \mathbb{N}_n$, and $e_j \in \mathbb{R}^n$ is the j-th coordinate vector.

2 Method

To solve (1.1), we use an accelerated gradient-projection method. Given a iterate $x_k \in \mathcal{X}$, an improvement x_{k+1} is found as follows:

1. Stop if x_k satisfies suitable termination criteria.

2. Compute
$$x_k^{\mathrm{C}} = P_{\mathcal{X}}[x_k - \alpha_k g(x_k)]$$
for some suitable α_k for which $f(x_k^{\mathrm{C}})) < f(x_k)$.

3. Compute x_k^{S} as an approximation to
$$\arg\min_{x\in\mathbb{R}^n} f(x) \text{ subject to } x_j = x_l \text{ for } j \in \mathcal{B}^{\mathrm{L}}(x_k^{\mathrm{C}}) \text{ and } x_j = x_u \text{ for } j \in \mathcal{B}^{\mathrm{U}}(x_k^{\mathrm{C}})$$

4. Compute
$$x_{k+1} = P_{\mathcal{X}}[x_k^{\mathrm{C}} + \beta_k(x_k^{\mathrm{S}} - x_k^{\mathrm{C}})]$$
for some suitable β_k for which $f(x_{k+1}) \leq f(x_k^{\mathrm{C}})$.

Steps 3 and 4 may sometimes be omitted if good progress is made in Step 2.

The dominant computations involved are the piecewise linesearches in Steps 2 and 4, and the linear least-squares minimization with fixed variables in Step 3. We consider each in turn.

2.1 Piecewise linesearches

Given a base point x^{S} and a search direction d, consider the path
$$x(\alpha) := P_{\mathcal{X}}(x^{\mathrm{S}} + \alpha d)$$
for $\alpha \geq 0$. On this path, our aim is to find a point for which $f(x(\alpha))$ is "suitably" smaller than $f(x^{\mathrm{S}})$. Clearly $x(\alpha)$ is piecewise linear, and changes direction at a finite sequence of "breakpoints" $\alpha_i > \alpha_{i-1}$, for $i = 1, \ldots m$, with $\alpha_0 = 0$. At breakpoint α_i, one or more of the variables $[x]_j$, $j \in \mathcal{B}_i$, encounters a bound, and each is fixed at that value for $\alpha \geq \alpha_i$. In particular,
$$x(\alpha_i + \Delta\alpha) = x_i + \Delta\alpha d_i \tag{2.2}$$
for all $\Delta\alpha \in [0, \Delta\alpha_i]$, where
$$x_i := x(\alpha_i) \equiv P_{\mathcal{X}}(x^{\mathrm{S}} + \alpha_i d),$$
$$[d_i]_j := \begin{cases} 0 & \text{if } \ j \in \bigcup_{k=0}^{i} \mathcal{B}_k \ \text{ or} \\ [d]_j & \text{otherwise,} \end{cases} \tag{2.3}$$
$$\Delta\alpha_i := \alpha_{i+1} - \alpha_i, \tag{2.4}$$
and specifically
$$\mathcal{B}_0 := \{j : [x^{\mathrm{S}}]_j = [x^{\mathrm{L}}]_j \ \text{ and } \ [d]_j < 0 \ \text{ or } \ [x^{\mathrm{S}}]_j = [x^{\mathrm{U}}]_j \ \text{ and } \ [d]_j > 0 \ \text{ or } \ [d]_j = 0\}.$$

While it might appear that the breakpoints need be sorted in advance, this is not necessary. Let

$$\alpha_j^{\mathrm{B}} := \begin{cases} \dfrac{[x^{\mathrm{U}}]_j - [x^{\mathrm{S}}]_j}{[d]_j} & \text{if } [d]_j > 0, \\ \dfrac{[x^{\mathrm{L}}]_j - [x^{\mathrm{S}}]_j}{[d]_j} & \text{if } [d]_j < 0 \text{ and} \\ 0 & \text{if } [d]_j = 0 \end{cases} \tag{2.5}$$

for $j \in \mathbb{N}_n$ be the unordered breakpoints. Then the i-th ordered breakpoint may be found efficiently knowing the $i-1$-st by arranging the $\{\alpha_j^{\mathrm{B}}\}$ in a heap, and using the Heapsort method [10]. In practice, breakpoints that are very close together are considered as a single point.

2.1.1 An exact piecewise minimizer

We now consider $f(x)$ on the segment (2.2). It follows immediately that

$$\begin{aligned} f(x(\alpha_i + \Delta\alpha)) &= f(x_i + \Delta\alpha d_i) = \tfrac{1}{2}\|A(x_i + \Delta\alpha d_i) - b\|^2 \\ &= \tfrac{1}{2}\|r_i + \Delta\alpha s_i\|^2 \\ &= \tfrac{1}{2}\|r_i\|^2 + \Delta\alpha\langle r_i, s_i\rangle + \tfrac{1}{2}\Delta\alpha^2\|s_i\|^2, \end{aligned} \tag{2.6}$$

where

$$r_i := Ax_i - b \tag{2.7}$$

and

$$s_i := Ad_i. \tag{2.8}$$

Notice that (2.6) implies that $f(x(\alpha))$ is convex, and that it achieves its smallest value at

$$\alpha_i^{\mathrm{M}} := \alpha_i - \frac{\langle r_i, s_i\rangle}{\|s_i\|^2}$$

so long as this is lies in the interval $[\alpha_i, \alpha_{i+1}]$. Thus an obvious method to find the global minimum on $x(\alpha)$ is to step through the breakpoints α_i in increasing order until either $\langle r_i, s_i\rangle \geq 0$, in which case the minimizer occurs at $\alpha = \alpha_i$ or $\alpha_i^{\mathrm{M}} \leq \alpha_{i+1}$, in which case a minimizer is at $\alpha = \alpha_i^{\mathrm{M}}$ [2, §3], [3, Alg.17.3.1 with typo corrections]. The process is finite, since there are only at most $n+1$ breakpoints. As we shall now show, the whole process can be implemented while moving from one segment to the next by evaluating and using the product Av, where v is a vector whose non-zeros occur only in positions corresponding to components of x that reach bounds as the segment ends; typically v is extremely sparse, usually only having a single nonzero.

Superficially, the scheme suggested above requires that we calculate the slope and curvature

$$f_i' := \langle r_i, s_i\rangle \quad \text{and} \quad f_i'' := \|s_i\|^2,$$

but as we shall now see these may be recurred with modest expense rather than formed afresh. The key is the definition (2.3) of d_i. In particular

$$d_{i+1} = d_i - v_{i+1}, \tag{2.9}$$

where

$$v_{i+1} := \sum_{j \in \mathcal{B}_{i+1}} [d]_j e_j,$$

and v_{i+1} is almost certainly a very sparse vector. Thus since A is sparse, it is also highly likely that so is

$$p_{i+1} := A v_{i+1}; \tag{2.10}$$

each column of A can only be accessed a single time during the entire iteration. Hence, using (2.8), (2.9) and (2.10),

$$s_{i+1} = s_i - p_{i+1} \tag{2.11}$$

differs from s_i in only a few components and therefore may be updated efficiently rather than recomputed. Therefore

$$f''_{i+1} = \|s_{i+1}\|^2 = \|s_i - p_{i+1}\|^2 = \|s_i\|^2 + \langle p_{i+1} - 2s_i, p_{i+1}\rangle = f''_i + \langle p_{i+1} - 2s_i, p_{i+1}\rangle$$

which may be updated using a sparse inner product involving p_{i+1}.

Recurring the slope $f'_i = \langle r_i, s_i\rangle$ is only slightly more awkward. To proceed, let

$$\begin{aligned} r^{\mathrm{s}} &:= A x^{\mathrm{s}} - b, & (2.12)\\ g^{\mathrm{s}} &:= A^T r^{\mathrm{s}}, & (2.13)\\ \Delta x_i &:= x_i - x^{\mathrm{s}} \;\text{ and} & (2.14)\\ q_i &:= A \Delta x_i. & (2.15)\end{aligned}$$

It then follows from (2.7), (2.12), (2.14) and (2.15) that

$$r_i = A(x^{\mathrm{s}} + \Delta x_i) - b = A x^{\mathrm{s}} - b + A\Delta x_i = A x^{\mathrm{s}} - b + q_i = r^{\mathrm{s}} + q_i,$$

and hence from (2.8) and (2.13) that

$$\langle r_i, s_i\rangle = \langle r^{\mathrm{s}} + q_i, s_i\rangle = \langle r^{\mathrm{s}}, s_i\rangle + \langle q_i, s_i\rangle = l_i + \gamma_i,$$

where

$$l_i := \langle g^{\mathrm{s}}, d_i\rangle \;\text{ and }\; \gamma_i := \langle q_i, s_i\rangle. \tag{2.16}$$

Thus (2.9) and (2.16) give

$$l_{i+1} = \langle g^{\mathrm{s}}, d_{i+1}\rangle = \langle g^{\mathrm{s}}, d_i - v_{i+1}\rangle = \langle g^{\mathrm{s}}, d_i\rangle - \langle g^{\mathrm{s}}, v_{i+1}\rangle = l_i - \langle g^{\mathrm{s}}, v_{i+1}\rangle = l_i - \langle r^{\mathrm{s}}, p_{i+1}\rangle,$$

and so l_{i+1} may be updated from l_i using a sparse inner product involving either v_{i+1} or p_{i+1}. Since (2.2) gives

$$x_{i+1} = x_i + \Delta\alpha_i d_i,$$

it follows from (2.8), (2.14) and (2.15) that

$$\begin{aligned} q_{i+1} &= A\Delta x_{i+1} = A(x_{i+1} - x^{\mathrm{s}}) = A(x_i + \Delta\alpha_i d_i - x^{\mathrm{s}})\\ &= A(x_i - x^{\mathrm{s}}) + \Delta\alpha_i A d_i = A\Delta x_i + \Delta\alpha_i A d_i \qquad (2.17)\\ &= q_i + \Delta\alpha_i s_i. \end{aligned}$$

Combining this with (2.11), we find

$$\begin{aligned}\gamma_{i+1} &= \langle q_{i+1}, s_{i+1}\rangle = \langle q_i + \Delta\alpha_i s_i, s_i - p_{i+1}\rangle = \langle q_i, s_i\rangle + \Delta\alpha_i \langle s_i, s_i\rangle - \langle q_{i+1}, p_{i+1}\rangle \\ &= \gamma_i + \Delta\alpha_i f_i'' - \langle q_{i+1}, p_{i+1}\rangle,\end{aligned} \tag{2.18}$$

where we only need to take the inner product over components j for which $[p_{i+1}]_j \neq 0$. Unfortunately, the update (2.17) for q_{i+1} does not involve a sparse vector, and we need to dig a little deeper. The secret is to define

$$u_{i+1} := q_{i+1} - \alpha_{i+1} s_i, \quad \text{with} \quad u_0 = 0. \tag{2.19}$$

Then it follows from (2.17), $\Delta\alpha_i - \alpha_{i+1} = -\alpha_i$ from (2.4), and (2.11) that

$$\begin{aligned}u_{i+1} &= q_i + \Delta\alpha_i\, s_i - \alpha_{i+1} s_i = q_i + (\Delta\alpha_i - \alpha_{i+1}) s_i = q_i - \alpha_i s_i = q_i - \alpha_i (s_{i-1} - p_i) \\ &= u_i + \alpha_i p_i,\end{aligned}$$

and this is a sparse update. Thus rather than recurring q_i, we may instead recur u_i and obtain $q_{i+1} = u_{i+1} + \alpha_{i+1} s_i$ from (2.19) as required. The important difference is that the recursions for u_{i+1} and s_i only involve the likely-sparse p_i. In particular, the recurrence for γ_{i+1} in (2.18) becomes

$$\gamma_{i+1} = \gamma_i + \Delta\alpha_i f_i'' - \langle u_{i+1}, p_{i+1}\rangle - \alpha_{i+1}\langle s_i, p_{i+1}\rangle.$$

We summarize our findings as Algorithm 2.1; this is essentially a specific case of a more general framework [5, Alg.4]. Notice that we also record the value $f_i = f(x_i)$ as a bi-product, and that this may be updated using (2.6) as we proceed from breakpoint to breakpoint.

Algorithm 2.1: Finding the piecewise arc minimizer $x^{\mathrm{C}} = P_{\mathcal{X}}(x^{\mathrm{S}} + \alpha^{\mathrm{C}} d)$ of f

0. Initialization: The initial point $x^{\mathrm{S}} \in \mathcal{X}$ and search direction d are given. Compute the residual $r^{\mathrm{S}} = Ax^{\mathrm{S}} - b$, and the breakpoints α_j^{B} from (2.5) for all $j \in \mathbb{N}_n$. Let $\alpha_0 = 0$, $u_0 = 0$,

$$\mathcal{B}_0 = \{j \in \mathbb{N}_n : \alpha_j^{\mathrm{B}} = 0\} \quad \text{and} \quad d_0 = d - \sum_{j \in \mathcal{B}_0} [d]_j e_j,$$

compute

$$\begin{aligned}&p_0 = Ad_0, \\ &f_0 = \tfrac{1}{2}\|r^{\mathrm{S}}\|^2, \quad f_0' = \langle r^{\mathrm{S}}, p_0\rangle \quad \text{and} \quad f_0'' = \|p_0\|^2,\end{aligned}$$

and set $s_0 = p_0$ and $i = 0$.

1. Find the next breakpoint: Determine α_{i+1}, the first breakpoint beyond α_i.

2. Check the current interval for arc minimizer:

If $f_i' \geq 0$, set $\alpha^{\mathrm{C}} = \alpha_i$, $x^{\mathrm{C}} = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha^{\mathrm{C}} d]$, $f(x^{\mathrm{C}}) = f_i$, and stop.

If $f_i'' > 0$ and $\alpha_i - f_i'/f_i'' \leq \alpha_{i+1}$, set $\alpha^{\mathrm{C}} = \alpha_i - f_i'/f_i''$, $x^{\mathrm{C}} = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha^{\mathrm{C}} d]$ $f(x^{\mathrm{C}}) = f_i + (\alpha^{\mathrm{C}} - \alpha_i) f_i' + \frac{1}{2}(\alpha^{\mathrm{C}} - \alpha_i)^2 f_i''$, and stop.

3. Prepare for the next interval: Set $\Delta\alpha_i = \alpha_{i+1} - \alpha_i$, recur

$$u_{i+1} = u_i + \alpha_i p_i,$$

let

$$\mathcal{B}_{i+1} = \{j \in \mathbb{N}_n : \alpha_j^{\mathrm{B}} = \alpha_{i+1}\} \;\text{ and }\; v_{i+1} = \sum_{j \in \mathcal{B}_{i+1}} [d]_j e_j,$$

and compute

$$p_{i+1} = A v_{i+1}.$$

4. Compute the value, slope and curvature: Compute

$$\begin{aligned} f_{i+1} &= f_i + \Delta\alpha_i f_i' + \tfrac{1}{2}(\Delta\alpha_i)^2 f_i'', \\ f_{i+1}' &= f_i' + \Delta\alpha_i f_i'' - \langle r^{\mathrm{S}} + u_{i+1} + \alpha_{i+1} s_i, p_{i+1} \rangle \;\text{ and} \\ f_{i+1}'' &= f_i'' + \langle p_{i+1} - 2 s_i, p_{i+1} \rangle, \end{aligned}$$

update

$$s_{i+1} = s_i - p_{i+1},$$

increment i by 1 and return to Step 1.

7In practice, we compute f_{i+1}' afresh when $|f_{i+1}'/f_i'|$ becomes small to guard against possible accumulated rounding errors in the recurrences. An earlier variant [1] based on algorithms for general quadratic objectives [2, §3], but specialised for the least-squares case, required products with both A and A^T at each breakpoint.

2.1.2 An approximate piecewise minimizer

In some cases, it may instead be advantageous to approximate the piecewise minimizer using a safeguarded, backtracking, piecewise linesearch [8]. The idea is simply to pick an initial stepsize α_0, backtracking reduction factor $\beta \in (0, 1)$ and decrease tolerance $\eta \in (0, \frac{1}{2})$, and to choose a sequence of decreasing stepsizes $\alpha_i := \alpha_0 \beta^i$ for increasing $i \geq 0$ until

$$f(x_i) \leq f(x^{\mathrm{S}}) + \eta \langle d_i, g^{\mathrm{S}} \rangle, \tag{2.20}$$

involving the trial point $x_i = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha_i d]$, the trial step $d_i = x_i - x^{\mathrm{S}}$, and the gradient at the base point $g^{\mathrm{S}} = A^T(Ax^{\mathrm{S}} - b)$.

To do so, we take advantage of the structure of the trial step d_i and basic properties of the backtracking projected line search. In particular, we know that once a component, say the jth, satisfies $[x_i^{\mathrm{L}}]_j < [x_i]_j < [x_i^{\mathrm{U}}]_j$, then $[x_\ell^{\mathrm{L}}]_j < [x_\ell]_j < [x_\ell^{\mathrm{U}}]_j$ will continue to hold for all $\ell \geq i$. Thus, by contrast to the search for the exact minimizer in Section 2.1.1 that moves forward along the piecewise projected gradient path fixing variables, the search here frees variables from their bounds as it proceeds backwwards towards x^{S}.

With this in mind, we compute the index set

$$\mathcal{A}^{\mathrm{S}} = \{j : [d]_j = 0 \;\text{ or }\; [x^{\mathrm{S}}]_j = [x^{\mathrm{L}}]_j \text{ and } [d]_j < 0 \;\text{ or }\; [x^{\mathrm{S}}]_j = [x^{\mathrm{U}}]_j \text{ and } [d]_j > 0\} \tag{2.21}$$

of components that are fixed at the base point on the piecewise search arc, record the vector x^{B} for which

$$[x^{\mathrm{B}}]_j := \begin{cases} [x^{\mathrm{S}}]_j & \text{if } j \in \mathcal{A}^{\mathrm{S}}, \\ [x^{\mathrm{L}}]_j & \text{if } j \notin \mathcal{A}^{\mathrm{S}} \text{ and } [d]_j < 0, \text{ and} \\ [x^{\mathrm{U}}]_j & \text{if } j \notin \mathcal{A}^{\mathrm{S}} \text{ and } [d]_j > 0, \end{cases} \tag{2.22}$$

and maintain the sets of active (i.e., fixed) and free components

$$\mathcal{A}_i := \{j \notin \mathcal{A}^{\mathrm{S}} : [x_i]_j = [x^{\mathrm{B}}]_j\} \;\text{ and} \tag{2.23}$$

$$\mathcal{F}_i := \{j \notin \mathcal{A}^{\mathrm{S}} : [x_i]_j \neq [x^{\mathrm{B}}]_j\}, \tag{2.24}$$

for $i \geq 0$, as well as the intermediate components, those that change from active to free,

$$\mathcal{I}_i := \mathcal{F}_i \cap \mathcal{A}_{i-1}, \tag{2.25}$$

for $i \geq 1$, at x_i. As we have already mentioned,

$$\mathcal{F}_i \subseteq \mathcal{F}_{i+1} \;\text{ and }\; \mathcal{A}_{i+1} \subseteq \mathcal{A}_i \quad \text{for all } i \geq 0$$

as a consequence of the approximate piecewise search.

It is immediate that, permuting the variables in the obvious way,

$$d_i = \begin{pmatrix} s_{\mathcal{A}_i} \\ \alpha_i d_{\mathcal{F}_i} \end{pmatrix}, \tag{2.26}$$

where

$$s := x^{\mathrm{B}} - x^{\mathrm{S}}.$$

Hence

$$r_i := Ax_i - b = A(x^{\mathrm{S}} + d_i) - b = r^{\mathrm{S}} + Ad_i = r^{\mathrm{S}} + A_{\mathcal{A}_i} s_{\mathcal{A}_i} + \alpha_i A_{\mathcal{F}_i} d_{\mathcal{F}_i} = r_i^{\mathrm{A}} + \alpha_i r_i^{\mathrm{F}},$$

where

$$r_i^{\mathrm{A}} := r^{\mathrm{S}} + A_{\mathcal{A}_i} s_{\mathcal{A}_i} \;\text{ and }\; r_i^{\mathrm{F}} := A_{\mathcal{F}_i} d_{\mathcal{F}_i}, \tag{2.27}$$

and therefore

$$f(x_i) = \tfrac{1}{2}\|r_i^{\mathrm{A}}\|^2 + \alpha_i \langle r_i^{\mathrm{A}}, r_i^{\mathrm{F}} \rangle + \tfrac{1}{2}\alpha_i^2 \|r_i^{\mathrm{F}}\|^2.$$

Thus $f(x_i)$ may be found trivially from the three quantities

$$f_i^{\mathrm{C}} := \tfrac{1}{2}\|r_i^{\mathrm{A}}\|^2, \;\; f_i^{\mathrm{L}} := \langle r_i^{\mathrm{A}}, r_i^{\mathrm{F}} \rangle \;\text{ and }\; f_i^{\mathrm{Q}} := \tfrac{1}{2}\|r_i^{\mathrm{F}}\|^2. \tag{2.28}$$

Although it is possible to compute these quantities afresh, it is usually more efficient to recur them as the piecewise linesearch proceeds instead. To do so, note that

$$\mathcal{A}_i = \mathcal{A}_{i-1} \setminus \mathcal{I}_i \;\text{ and }\; \mathcal{F}_i = \mathcal{F}_{i-1} \cup \mathcal{I}_i, \tag{2.29}$$

and thus that

$$r_i^{\mathrm{A}} = r_{i-1}^{\mathrm{A}} - p_i \;\text{ and }\; r_i^{\mathrm{F}} = r_{i-1}^{\mathrm{F}} + q_i, \tag{2.30}$$

where

$$p_i := A_{\mathcal{I}_i} s_{\mathcal{I}_i} \;\text{ and }\; q_i := A_{\mathcal{I}_i} d_{\mathcal{I}_i}. \tag{2.31}$$

Therefore, when A is sparse and if a modest number of variables move off bounds at x_i, the vectors p_i and q_i will most likely be sparse, indeed, aside from exact cancellations, their nonzeros will occur in the same positions. It then follows from (2.28) and (2.31) that

$$\begin{aligned} f_i^{\mathrm{C}} &= f_{i-1}^{\mathrm{C}} - \langle p_i, r_{i-1}^{\mathrm{A}} \rangle + \tfrac{1}{2}\|p_i\|^2, \\ f_i^{\mathrm{L}} &= f_{i-1}^{\mathrm{L}} + \langle q_i, r_{i-1}^{\mathrm{A}} \rangle - \langle p_i, r_{i-1}^{\mathrm{F}} \rangle - \langle q_i, p_i \rangle \;\text{ and} \\ f_i^{\mathrm{Q}} &= f_{i-1}^{\mathrm{Q}} + \langle q_i, r_{i-1}^{\mathrm{F}} \rangle + \tfrac{1}{2}\|q_i\|^2, \end{aligned} \tag{2.32}$$

involving sparse inner products.

In order to check (2.20), we also need to compute

$$\langle d_i, g^{\mathrm{S}} \rangle = \langle A^T d_i, r^{\mathrm{S}} \rangle = \langle A_{\mathcal{A}_i} s_{\mathcal{A}_i}, r^{\mathrm{S}} \rangle + \alpha_i \langle A_{\mathcal{F}_i} d_{\mathcal{F}_i}, r^{\mathrm{S}} \rangle = \gamma_i^{\mathrm{A}} + \alpha_i \gamma_i^{\mathrm{F}}, \tag{2.33}$$

where

$$\gamma_i^{\mathrm{A}} := \langle A_{\mathcal{A}_i} s_{\mathcal{A}_i}, r^{\mathrm{S}} \rangle \;\text{ and }\; \gamma_i^{\mathrm{F}} := \langle A_{\mathcal{F}_i} d_{\mathcal{F}_i}, r^{\mathrm{S}} \rangle,$$

using (2.26). It then follows from (2.29) and (2.31) that

$$\begin{aligned} \gamma_i^{\mathrm{A}} &= \langle A_{\mathcal{A}_{i-1}} s_{\mathcal{A}_{i-1}}, r^{\mathrm{S}} \rangle - \langle A_{\mathcal{I}_i} s_{\mathcal{I}_i}, r^{\mathrm{S}} \rangle = \gamma_{i-1}^{\mathrm{A}} - \langle p_i, r^{\mathrm{S}} \rangle \;\text{ and} \\ \gamma_i^{\mathrm{F}} &= \langle A_{\mathcal{F}_{i-1}} d_{\mathcal{F}_{i-1}}, r^{\mathrm{S}} \rangle + \langle A_{\mathcal{I}_i} d_{\mathcal{I}_i}, r^{\mathrm{S}} \rangle = \gamma_{i-1}^{\mathrm{F}} + \langle q_i, r^{\mathrm{S}} \rangle. \end{aligned} \tag{2.34}$$

Thus we may obtain (2.33) by recurring γ_i^{A} and γ_i^{F} using (2.34), and the latter simply requires a further pair of sparse inner products.

It is important to notice that the crucial likely-sparse vectors p_i and q_i in (2.31) needed by the recurrences (2.32) and (2.34) only depend on the set of indices $\mathcal{I}_i$ that change status from fixed to free during the i-th backtrack. Although formally we define this using x_i, in practice we do not need to form x_i, indeed to do so would require a projection $P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha_i d]$ for each attempted step α_i. Instead, just as in Section 2.1.1, we find the unordered breakpoints (2.5), and then arrange them in a "backward" heap starting at the first for which $\alpha_j^{\mathrm{B}} > \alpha_0$. We then adjust the heap to find precisely the indices I_{i+1} of those between α_i and α_{i+1} using the Heapsort algorithm as required.

We summarize our discussion as Algorithm 2.2 on the next page.

Algorithm 2.2: Find an approximate backtracking arc minimizer $\boldsymbol{x}^{\mathrm{C}}$ of $\boldsymbol{f}$

0. Initialization: The initial point $x^{\mathrm{S}} \in \mathcal{X}$, search direction d, initial stepsize $\alpha_0 > 0$, reduction factor $\beta \in (0,1)$ and decrease tolerance $\eta \in (0, \frac{1}{2})$ are given. Compute the residual $r^{\mathrm{S}} = Ax^{\mathrm{S}} - b$, the initial objective value $f(x^{\mathrm{S}}) = \frac{1}{2}\|r^{\mathrm{S}}\|^2$, the base fixed set $\mathcal{A}^{\mathrm{S}}$ from (2.21), the breakpoints α_j^{B} from (2.5) for all $j \in \mathbb{N}_n$, the end of the arc x^{B} from (2.22) and its direction $s = x^{\mathrm{B}} - x^{\mathrm{S}}$, the initial search point $x_0 = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha_0 d]$, the active and free components at x_0,

$$\mathcal{A}_0 = \{j : \alpha_0 > \alpha_j^{\mathrm{B}}\} \text{ and } \mathcal{F}_0 = \{j \notin \mathcal{A}^{\mathrm{S}} : \alpha_0 \le \alpha_j^{\mathrm{B}}\},$$

and the corresponding residuals

$$r_0^{\mathrm{A}} = r^{\mathrm{S}} + p_0 \text{ and } r_0^{\mathrm{F}} = q_0$$

using the matrix-vector products

$$p_0 = A_{\mathcal{A}_0} s_{\mathcal{A}_0} \text{ and } q_0 = A_{\mathcal{F}_0} d_{\mathcal{F}_0}.$$

Initialize

$$f_0^{\mathrm{C}} = \tfrac{1}{2}\|r_0^{\mathrm{A}}\|^2, \;\; f_0^{\mathrm{L}} = \langle r_0^{\mathrm{A}}, r_0^{\mathrm{F}} \rangle \text{ and } f_0^{\mathrm{Q}} = \tfrac{1}{2}\|r_0^{\mathrm{F}}\|^2,$$

as well as

$$\gamma_0^{\mathrm{A}} = \langle p_0, r^{\mathrm{S}} \rangle \text{ and } \gamma_0^{\mathrm{F}} = \langle q_0, r^{\mathrm{S}} \rangle.$$

Set $i = 0$.

1. Check for an approximate arc minimizer: Compute

$$f_i = f_i^{\mathrm{C}} + \alpha_i f_i^{\mathrm{L}} + \alpha_i^2 f_i^{\mathrm{Q}} \text{ and } \gamma_i = \gamma_i^{\mathrm{A}} + \alpha_i \gamma_1^{\mathrm{F}}.$$

If $f_i \le f(x^{\mathrm{S}}) + \eta\gamma_i$, set

$$\alpha^{\mathrm{C}} = \alpha_i, \;\; x^{\mathrm{C}} = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha^{\mathrm{C}} d] \text{ and } f(x^{\mathrm{C}}) = f_i,$$

and stop.

2. Find the next set of indices that change status: Let $\alpha_{i+1} = \beta\alpha_i$, and compute

$$\mathcal{I}_{i+1} = \{j : \alpha_{i+1} < \alpha_j^{\mathrm{B}} \le \alpha_i\}$$

using the Heapsort algorithm.

3. Update the components of the objective and its slope: Compute

$$p_{i+1} = A_{\mathcal{I}_{i+1}} s_{\mathcal{I}_{i+1}} \text{ and } q_{i+1} = A_{\mathcal{I}_{i+1}} d_{\mathcal{I}_{i+1}},$$

update

$$
\begin{aligned}
f^{\mathrm{C}}_{i+1} &= f^{\mathrm{C}}_i - \langle p_{i+1}, r^{\mathrm{A}}_i \rangle + \tfrac{1}{2}\|p_{i+1}\|^2,\\
f^{\mathrm{L}}_{i+1} &= f^{\mathrm{L}}_i + \langle q_{i+1}, r^{\mathrm{A}}_i \rangle - \langle p_{i+1}, r^{\mathrm{F}}_i \rangle - \langle q_{i+1}, p_{i+1} \rangle,\\
f^{\mathrm{Q}}_{i+1} &= f^{\mathrm{Q}}_i + \langle q_{i+1}, r^{\mathrm{F}}_i \rangle + \tfrac{1}{2}\|q_{i+1}\|^2,\\
\gamma^{\mathrm{A}}_{i+1} &= \gamma^{\mathrm{A}}_i - \langle p_{i+1}, r^{\mathrm{S}} \rangle,\\
\gamma^{\mathrm{F}}_{i+1} &= \gamma^{\mathrm{F}}_i + \langle q_{i+1}, r^{\mathrm{S}} \rangle,\\
r^{\mathrm{A}}_{i+1} &= r^{\mathrm{A}}_i - p_{i+1} \;\text{ and}\\
r^{\mathrm{F}}_{i+1} &= r^{\mathrm{F}}_i + q_{i+1},
\end{aligned}
$$

increment i by 1 and return to Step 1.

It is also possible to contemplate a variant in which the iterates advances further along the piecewise arc if the initial point x_0 is acceptable. To be specific, if

$$f(x_0) \leq f(x^{\mathrm{S}}) + \eta\langle x_0 - x^{\mathrm{S}}, g^{\mathrm{S}} \rangle, \tag{2.35}$$

we terminate at the arc point x_i with the smallest $i \geq 0$ for which

$$f(x_{i+1}) > f(x^{\mathrm{S}}) + \eta\langle x_{i+1} - x^{\mathrm{S}}, g^{\mathrm{S}} \rangle, \tag{2.36}$$

where, as before, $x_i = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha_i d]$, but now $\alpha_i = \alpha_0 \beta^{-i} \geq \alpha_0$. The only essential difference is that in this case

$$\mathcal{F}_{i+1} \subseteq \mathcal{F}_i \;\text{ and }\; \mathcal{A}_i \subseteq \mathcal{A}_{i+1} \quad \text{for all } i \geq 0 \;\;,$$

and components in $\mathcal{J}_{i+1} := \mathcal{A}_{i+1} \cap \mathcal{F}_i$ change from free to active, i.e.,

$$\mathcal{A}_{i+1} = \mathcal{A}_i \cup \mathcal{J}_{i+1} \;\text{ and }\; \mathcal{F}_{i+1} = \mathcal{F}_i \setminus \mathcal{J}_{i+1}. \tag{2.37}$$

This leads to

$$r^{\mathrm{A}}_{i+1} = r^{\mathrm{A}}_i + p_{i+1} \;\text{ and }\; r^{\mathrm{F}}_{i+1} = r^{\mathrm{F}}_i - q_{i+1},$$

where p_i and q_i are given by (2.31), and the obvious changes to (2.32) and (2.34). We summarize the necessary enhancements in Algorithm 2.3.

Algorithm 2.3: Find an approximate piecewise arc minimizer $\boldsymbol{x^{\mathrm{C}}}$ of $\boldsymbol{f}$

The same as Algorithm 2.2 on the previous page except that Step 1 becomes

1. Check for an approximate backtracking arc minimizer: Compute

$$f_i = f^{\mathrm{C}}_i + \alpha_i f^{\mathrm{L}}_i + \alpha_i^2 f^{\mathrm{Q}}_i \;\text{ and }\; \gamma_i = \gamma^{\mathrm{A}}_i + \alpha_i \gamma^{\mathrm{F}}_1.$$

If $f_i \leq f(x^{\mathrm{S}}) + \eta\gamma_i$, go to Step 4 if $i = 0$ but otherwise, i.e., if $i > 0$, set

$$\alpha^{\mathrm{C}} = \alpha_i, \;\; x^{\mathrm{C}} = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha^{\mathrm{C}} d] \;\text{ and }\; f(x^{\mathrm{C}}) = f_i,$$

and stop.

and additionally

4. Find the next set of indices that change status: Let $\alpha_{i+1} = \beta^{-1}\alpha_i$, and compute

$$\mathcal{J}_{i+1} = \{j : \alpha_i < \alpha_j^{\mathrm{B}} \le \alpha_{i+1}\}$$

using the Heapsort algorithm.

5. Update the components of the objective and its slope: Compute

$$p_{i+1} = A_{\mathcal{J}_{i+1}} s_{\mathcal{J}_{i+1}} \text{ and } q_{i+1} = A_{\mathcal{J}_{i+1}} d_{\mathcal{J}_{i+1}},$$

update

$$\begin{aligned}
f_{i+1}^{\mathrm{C}} &= f_i^{\mathrm{C}} + \langle p_{i+1}, r_i^{\mathrm{A}} \rangle + \tfrac{1}{2}\|p_{i+1}\|^2, \\
f_{i+1}^{\mathrm{L}} &= f_i^{\mathrm{L}} - \langle q_{i+1}, r_i^{\mathrm{A}} \rangle + \langle p_{i+1}, r_i^{\mathrm{F}} \rangle - \langle q_{i+1}, p_{i+1} \rangle, \\
f_{i+1}^{\mathrm{Q}} &= f_i^{\mathrm{Q}} - \langle q_{i+1}, r_i^{\mathrm{F}} \rangle + \tfrac{1}{2}\|q_{i+1}\|^2, \\
\gamma_{i+1}^{\mathrm{A}} &= \gamma_i^{\mathrm{A}} + \langle p_{i+1}, r^{\mathrm{S}} \rangle, \\
\gamma_{i+1}^{\mathrm{F}} &= \gamma_i^{\mathrm{F}} - \langle q_{i+1}, r^{\mathrm{S}} \rangle, \\
r_{i+1}^{\mathrm{A}} &= r_i^{\mathrm{A}} + p_{i+1} \text{ and} \\
r_{i+1}^{\mathrm{F}} &= r_i^{\mathrm{F}} - q_{i+1}.
\end{aligned}$$

6. Check for an approximate extended arc minimizer: Compute

$$f_{i+1} = f_{i+1}^{\mathrm{C}} + \alpha_{i+1} f_{i+1}^{\mathrm{L}} + \alpha_{i+1}^2 f_{i+1}^{\mathrm{Q}} \text{ and } \gamma_{i+1} = \gamma_{i+1}^{\mathrm{A}} + \alpha_{i+1}\gamma_{1+1}^{\mathrm{F}}.$$

If $f_{i+1} > f(x^{\mathrm{S}}) + \eta\gamma_{i+1}$ or $\alpha_{i+1} \ge \max_j \alpha_j^{\mathrm{B}}$, set

$$\alpha^{\mathrm{C}} = \alpha_i, \;\; x^{\mathrm{C}} = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha^{\mathrm{C}} d] \text{ and } f(x^{\mathrm{C}}) = f_i,$$

and stop. Otherwise, increment i by 1 and return to Step 4.

Notice the extra stopping check in Step 6: this is simply to prevent the search extending beyond the end of the piecewise arc.

2.2 Linear least-squares minimization with fixed variables

Let $Z[v]$ be the operator that sets the components of a given vector v to zero, i.e.,

$$[Z[v]]_i = \begin{cases} 0 & i \in \mathcal{Z} \\ [v]_i & \text{otherwise,} \end{cases}$$

for a specified subset $\mathcal{Z} \subseteq \mathbb{N}_n$ of indices of x that are to be fixed; in the context of Step 3 in the generic framework described at the start of Section 2, $\mathcal{Z}$ at iteration k will be $\mathcal{B}^{\mathrm{L}}(x_k^{\mathrm{C}}) \cup \mathcal{B}^{\mathrm{U}}(x_k^{\mathrm{C}})$. To minimize $f(x)$ over the set of variables that are in $\mathcal{F} := \mathbb{N}_n \setminus \mathcal{Z}$, while fixing the remaining components at the values that they have at x_0, we may apply the following well-known preconditioned conjugate-gradient iterative scheme [7, 9]—here the preconditioner M may be any symmetric, positive-definite matrix, for which the cost of solving $Mv = g$ is modest.

Algorithm 2.4: The preconditioned conjugate-gradient least-squares method

Given x_0, set $r_0 = Ax_0 - b$ and $g_0 = Z[A^T r_0]$, and let $v_0 = Z[M^{-1} g_0]$ and $p_0 = -v_0$. For $k = 0, 1, \ldots$ until convergence, perform the iteration

$$\begin{aligned} q_k &= Ap_k, \\ \alpha_k &= \langle g_k, v_k \rangle / \langle q_k, q_k \rangle, \\ x_{k+1} &= x_k + \alpha_k p_k, \\ r_{k+1} &= r_k + \alpha_k q_k, \\ g_{k+1} &= Z[A^T r_{k+1}], \\ v_{k+1} &= Z[M^{-1} g_{k+1}], \\ \beta_k &= \langle g_{k+1}, v_{k+1} \rangle / \langle g_k, v_k \rangle \ \text{ and} \\ p_{k+1} &= -v_{k+1} + \beta_k p_k. \end{aligned}$$

Notice that the product $q_k = Ap_k$ only requires access to the columns of A with indices that lie in $\mathcal{F}$. Likewise, only the components of $A^T r_{k+1}$ with indices that lie in $\mathcal{F}$ are needed. A good "preconditioner" M will be such that the eigenvalues of $M^{-1} A_{\mathcal{F}}^T A_{\mathcal{F}}$ are clustered around one, although it is not necessarily easy to achieve this [6]; at the very least, picking $M = \mathrm{diag}(A^T A)$ is often helpful, and in particular the required diagonal entries are then simply the squares of the norms of the columns of A.

3 Regularization

In practice, it is common to consider the regularized bound-constrained linear least-squares problem

$$\underset{x \in \mathcal{X}}{\text{minimize}} \ \ \phi(x, \sigma) := \tfrac{1}{2} \|Ax - b\|^2 + \tfrac{1}{2} \sigma \|x\|^2, \tag{3.38}$$

in which we allow an extra regularization term with weight $\sigma \geq 0$. Since $\phi(x, \sigma)$ is a linear sum of $f(x)$ and the regularization term, we may use linearity to compute derivatives involving those that we have already seen plus σ times those of $\rho(x) := \frac{1}{2}\|x\|^2$. Trivially, the gradient and Hessian of ρ are x and I respectively, where I is the n by n identity matrix. Equally $\rho(x)$ mat be interpreted as a sum-of-squares function $\frac{1}{2}\|Ax - b\|^2$ in the special case for which $A = I$ and $b = 0$. Such a perspective then simply allows us to derive the necessary changes in $\rho(x)$ as we investigate the piecewise-linear path $x(\alpha)$.

To generalise the method for finding the exact piecewise minimizer described in Section 2.1.1, we must consider $\rho(x)$ on the segment (2.2). Plainly we have that

$$\rho(x(\alpha_i + \Delta\alpha)) = \rho(x_i + \Delta\alpha d_i) = \tfrac{1}{2}\|x_i + \Delta\alpha d_i\|^2 = \rho_i + \Delta\alpha\rho_i' + \tfrac{1}{2}\Delta\alpha^2\rho_i'' \tag{3.39}$$

involving the value, slope and curvature

$$\rho_i := \tfrac{1}{2}\|x_i\|^2, \quad \rho_i' := \langle x_i, d_i\rangle \quad \text{and} \quad \rho_i'' := \|d_i\|^2$$

at breakpoint i. Mechanically repeating the arguments in Section 2.1.1, leads to the following generalisation of Algorithm 2.1 for the regularization case.

Algorithm 3.1: Finding the piecewise arc minimizer $\boldsymbol{x^{\mathrm{C}} = P_{\mathcal{X}}(x^{\mathrm{S}}+\alpha^{\mathrm{C}}d)}$ of $\boldsymbol{\phi}$

0. Initialization: The initial point $x^{\mathrm{S}} \in \mathcal{X}$ and search direction d are given. Compute the residual $r^{\mathrm{S}} = Ax^{\mathrm{S}} - b$, and the breakpoints α_j^{B} from (2.5) for all $j \in \mathbb{N}_n$. Let $\alpha_0 = 0$, $u_0 = 0$, $w_0 = 0$,

$$\mathcal{B}_0 = \{j \in \mathbb{N}_n : \alpha_j^{\mathrm{B}} = 0\} \quad \text{and} \quad d_0 = d - \sum_{j \in \mathcal{B}_0} [d]_j e_j,$$

compute

$$\begin{aligned} &\rho_0 = \tfrac{1}{2}\|x^{\mathrm{S}}\|^2, \quad \rho_0' = \langle x^{\mathrm{S}}, d_0\rangle \quad \text{and} \quad \rho_0'' = \|d_0\|^2, \\ &p_0 = Ad_0, \\ &f_0 = \tfrac{1}{2}\|r^{\mathrm{S}}\|^2, \quad f_0' = \langle r^{\mathrm{S}}, p_0\rangle \quad \text{and} \quad f_0'' = \|p_0\|^2, \end{aligned}$$

and set $v_0 = d_0$, $s_0 = p_0$ and $i = 0$.

1. Find the next breakpoint: Determine α_{i+1}, the first breakpoint beyond α_i.

2. Check the current interval for arc minimizer:

Compute

$$\phi_i = f_i + \sigma\rho_i, \quad \phi_i' = f_i' + \sigma\rho_i' \quad \text{and} \quad \phi_i'' = f_i'' + \sigma\rho_i''.$$

If $\phi_i' \geq 0$, set $\alpha^{\mathrm{C}} = \alpha_i$, $x^{\mathrm{C}} = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha^{\mathrm{C}} d]$, $\phi(x^{\mathrm{C}}, \sigma) = \phi_i$, and stop.

If $\phi_i'' > 0$ and $\alpha_i - \phi_i'/\phi_i'' \leq \alpha_{i+1}$, set $\alpha^{\mathrm{C}} = \alpha_i - \phi_i'/\phi_i''$, $x^{\mathrm{C}} = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha^{\mathrm{C}} d]$ $\phi(x^{\mathrm{C}}, \sigma) = \phi_i + (\alpha^{\mathrm{C}} - \alpha_i)\phi_i' + \frac{1}{2}(\alpha^{\mathrm{C}} - \alpha_i)^2\phi_i''$, and stop.

3. Prepare for the next interval: Set $\Delta\alpha_i = \alpha_{i+1} - \alpha_i$, recur

$$\begin{aligned} w_{i+1} &= w_i + \alpha_i v_i, \\ u_{i+1} &= u_i + \alpha_i p_i, \end{aligned}$$

let

$$\mathcal{B}_{i+1} = \{j \in \mathbb{N}_n : \alpha_j^{\text{B}} = \alpha_{i+1}\} \;\text{ and }\; v_{i+1} = \sum_{j \in \mathcal{B}_{i+1}} [d]_j e_j,$$

and compute

$$p_{i+1} = A v_{i+1}.$$

4. Compute the value, slope and curvature: Compute

$$\begin{aligned} \rho_{i+1} &= \rho_i + \Delta\alpha_i \rho_i' + \tfrac{1}{2}(\Delta\alpha_i)^2 \rho_i'', \\ \rho_{i+1}' &= \rho_i' + \Delta\alpha_i \rho_i'' - \langle x^{\text{S}} + w_{i+1} + \alpha_{i+1} d_i, v_{i+1} \rangle, \qquad (3.40) \\ \rho_{i+1}'' &= \rho_i'' + \langle v_{i+1} - 2d_i, v_{i+1} \rangle, \qquad (3.41) \\ f_{i+1} &= f_i + \Delta\alpha_i f_i' + \tfrac{1}{2}(\Delta\alpha_i)^2 f_i'', \\ f_{i+1}' &= f_i' + \Delta\alpha_i f_i'' - \langle r^{\text{S}} + u_{i+1} + \alpha_{i+1} s_i, p_{i+1} \rangle \;\text{ and} \\ f_{i+1}'' &= f_i'' + \langle p_{i+1} - 2s_i, p_{i+1} \rangle \end{aligned}$$

update

$$\begin{aligned} d_{i+1} &= d_i - v_{i+1}, \\ s_{i+1} &= s_i - p_{i+1}, \end{aligned}$$

increment i by 1 and return to Step 1.

Slightly less obviously, it is straightforward to show that

$$\langle d_i, v_{i+1} \rangle = \langle v_{i+1}, v_{i+1} \rangle,$$

and thus (3.40) and (3.41) may be written instead as

$$\rho_{i+1}' = \rho_i' + \Delta\alpha_i \rho_i'' - \langle x^{\text{S}} + w_{i+1} + \alpha_{i+1} v_{i+1}, v_{i+1} \rangle,$$

and

$$\rho_{i+1}'' = \rho_i'' - \langle v_{i+1}, v_{i+1} \rangle,$$

with no need to recur d_i in Step 4. Crucially, each of the recursions needed to maintain w_i, ρ_i, ρ_i' and ρ_i'' only requires operations involving the sparse vector v_i.

To adapt the method for finding an approximate piecewise minimizer described in Section 2.1.2 to cope with regularization, the only significant issue is to consider how the regularization term $\rho(x)$ evolves as we move backwards or forwards along the search arc.

Plainly we have that

$$x_i := x^{\mathrm{S}} + d_i = x^{\mathrm{S}} + s_{\mathcal{A}_i} + \alpha_i d_{\mathcal{F}_i} = x_i^{\mathrm{A}} + \alpha_i d_i^{\mathrm{F}},$$

where[1]

$$x_i^{\mathrm{A}} := x^{\mathrm{S}} + s_{\mathcal{A}_i} \text{ and } d_i^{\mathrm{F}} := d_{\mathcal{F}_i},$$

and therefore

$$\rho(x_i) = \rho_i^{\mathrm{C}} + \alpha_i \rho_i^{\mathrm{L}} + \alpha_i^2 \rho_i^{\mathrm{Q}},$$

where

$$\rho_i^{\mathrm{C}} := \tfrac{1}{2}\|x_i^{\mathrm{A}}\|^2, \;\; \rho_i^{\mathrm{L}} := \langle x_i^{\mathrm{A}}, d_i^{\mathrm{F}}\rangle = \langle x^{\mathrm{S}}, d_i^{\mathrm{F}}\rangle \text{ and } \rho_i^{\mathrm{Q}} := \tfrac{1}{2}\|d_i^{\mathrm{F}}\|^2.$$

It is then easy to simplify the recurrences described in the earlier section to deal with this. The generalisation of Algorithm 2.1 for the regularization case is then simply as follows.

Algorithm 3.2: Find an approximate backtracking arc minimizer $\boldsymbol{x}^{\mathbf{C}}$ of $\boldsymbol{\phi}$

0. Initialization: The initial point $x^{\mathrm{S}} \in \mathcal{X}$, search direction d, initial stepsize $\alpha_0 > 0$, reduction factor $\beta \in (0,1)$ and decrease tolerance $\eta \in (0, \frac{1}{2})$ are given. Compute the residual $r^{\mathrm{S}} = Ax^{\mathrm{S}} - b$, the initial objective value $f(x^{\mathrm{S}}) = \frac{1}{2}\|r^{\mathrm{S}}\|^2$, the base fixed set $\mathcal{A}^{\mathrm{S}}$ from (2.21), the breakpoints α_j^{B} from (2.5) for all $j \in \mathbb{N}_n$, the end of the arc x^{B} from (2.22) and its direction $s = x^{\mathrm{B}} - x^{\mathrm{S}}$, the initial search point $x_0 = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha_0 d]$, the active and free components at x_0,

$$\mathcal{A}_0 = \{j : \alpha_0 > \alpha_j^{\mathrm{B}}\} \text{ and } \mathcal{F}_0 = \{j \notin \mathcal{A}^{\mathrm{S}} : \alpha_0 \le \alpha_j^{\mathrm{B}}\},$$

and the corresponding values

$$x_0^{\mathrm{A}} = x^{\mathrm{S}} + s_{\mathcal{A}_0} \text{ and } d_0^{\mathrm{F}} = d_{\mathcal{F}_0}$$

and residuals

$$r_0^{\mathrm{A}} = r^{\mathrm{S}} + p_0 \text{ and } r_0^{\mathrm{F}} = q_0$$

using the matrix-vector products

$$p_0 = A_{\mathcal{A}_0} s_{\mathcal{A}_0} \text{ and } q_0 = A_{\mathcal{F}_0} d_{\mathcal{F}_0}.$$

Initialize

$$\begin{aligned} &f_0^{\mathrm{C}} = \tfrac{1}{2}\|r_0^{\mathrm{A}}\|^2, \;\; f_0^{\mathrm{L}} = \langle r_0^{\mathrm{A}}, r_0^{\mathrm{F}}\rangle \text{ and } f_0^{\mathrm{Q}} = \tfrac{1}{2}\|r_0^{\mathrm{F}}\|^2, \text{ and} \\ &\rho_0^{\mathrm{C}} = \tfrac{1}{2}\|x_0^{\mathrm{A}}\|^2, \;\; \rho_0^{\mathrm{L}} = \langle x^{\mathrm{S}}, d_0^{\mathrm{F}}\rangle \text{ and } \rho_0^{\mathrm{Q}} = \tfrac{1}{2}\|d_0^{\mathrm{F}}\|^2, \end{aligned}$$

as well as

$$\begin{aligned} &\gamma_0^{\mathrm{A}} = \langle r^{\mathrm{S}}, p_0\rangle \text{ and } \gamma_0^{\mathrm{F}} = \langle r^{\mathrm{S}}, q_0\rangle, \text{ and} \\ &\mu_0^{\mathrm{A}} = \langle x^{\mathrm{S}}, s_{\mathcal{A}_0}\rangle \text{ and } \mu_0^{\mathrm{F}} = \langle x^{\mathrm{S}}, d_{\mathcal{F}_0}\rangle. \end{aligned}$$

Set $i = 0$.

[1] Here $s_{\mathcal{A}_i}$ and $d_{\mathcal{F}_i}$ are considered as n-vectors, whose nonzero components are fed by the index sets and $\mathcal{F}_i$, respectively.

1. Check for an approximate arc minimizer: Compute

$$\begin{aligned} f_i &= f_i^{\mathrm{C}} + \alpha_i f_i^{\mathrm{L}} + \alpha_i^2 f_i^{\mathrm{Q}}, \\ \rho_i &= \rho_i^{\mathrm{C}} + \alpha_i \rho_i^{\mathrm{L}} + \alpha_i^2 \rho_i^{\mathrm{Q}}, \\ \phi_i &= f_i + \sigma \rho_i \;\text{ and} \\ \gamma_i &= \gamma_i^{\mathrm{A}} + \alpha_i \gamma_i^{\mathrm{F}} + \sigma(\mu_i^{\mathrm{A}} + \alpha_i \mu_i^{\mathrm{F}}). \end{aligned}$$

If $\phi_i \leq \phi(x^{\mathrm{S}}, \sigma) + \eta \gamma_i$, go to Step 4 if $i = 0$ but otherwise, i.e., if $i > 0$, set

$$\alpha^{\mathrm{C}} = \alpha_i, \;\; x^{\mathrm{C}} = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha^{\mathrm{C}} d], \;\; f(x^{\mathrm{C}}) = f_i \;\text{ and }\; \phi(x^{\mathrm{C}}, \sigma) = \phi_i,$$

and stop.

2. Find the next set of indices that change status: Let $\alpha_{i+1} = \beta \alpha_i$, and compute

$$\mathcal{I}_{i+1} = \{j : \alpha_{i+1} < \alpha_j^{\mathrm{B}} \leq \alpha_i\}$$

using the Heapsort algorithm.

3. Update the components of the objective and its slope: Compute

$$p_{i+1} = A_{\mathcal{I}_{i+1}} s_{\mathcal{I}_{i+1}} \;\text{ and }\; q_{i+1} = A_{\mathcal{I}_{i+1}} d_{\mathcal{I}_{i+1}}$$

update

$$\begin{aligned} f_{i+1}^{\mathrm{C}} &= f_i^{\mathrm{C}} - \langle r_i^{\mathrm{A}}, p_{i+1} \rangle + \tfrac{1}{2} \|p_{i+1}\|^2, \\ f_{i+1}^{\mathrm{L}} &= f_i^{\mathrm{L}} + \langle r_i^{\mathrm{A}}, q_{i+1} \rangle - \langle r_i^{\mathrm{F}}, p_{i+1} \rangle - \langle p_{i+1}, q_{i+1} \rangle, \\ f_{i+1}^{\mathrm{Q}} &= f_i^{\mathrm{Q}} + \langle r_i^{\mathrm{F}}, q_{i+1} \rangle + \tfrac{1}{2} \|q_{i+1}\|^2, \\ \gamma_{i+1}^{\mathrm{A}} &= \gamma_i^{\mathrm{A}} - \langle r^{\mathrm{S}}, p_{i+1} \rangle, \;\; \gamma_{i+1}^{\mathrm{F}} = \gamma_i^{\mathrm{F}} + \langle r^{\mathrm{S}}, q_{i+1} \rangle, \\ r_{i+1}^{\mathrm{A}} &= r_i^{\mathrm{A}} - p_{i+1} \;\text{ and }\; r_{i+1}^{\mathrm{F}} = r_i^{\mathrm{F}} + q_{i+1}, \end{aligned}$$

as well as

$$\begin{aligned} \rho_{i+1}^{\mathrm{C}} &= \rho_i^{\mathrm{C}} - \langle x_i^{\mathrm{A}}, s_{\mathcal{I}_{i+1}} \rangle + \tfrac{1}{2} \|s_{\mathcal{I}_{i+1}}\|^2, \\ \rho_{i+1}^{\mathrm{L}} &= \rho_i^{\mathrm{L}} + \langle x_i^{\mathrm{A}}, d_{\mathcal{I}_{i+1}} \rangle - \langle d_i^{\mathrm{F}}, s_{\mathcal{I}_{i+1}} \rangle - \langle d_{\mathcal{I}_{i+1}}, s_{\mathcal{I}_{i+1}} \rangle, \\ \rho_{i+1}^{\mathrm{Q}} &= \rho_i^{\mathrm{Q}} + \langle d_i^{\mathrm{F}}, d_{\mathcal{I}_{i+1}} \rangle + \tfrac{1}{2} \|d_{\mathcal{I}_{i+1}}\|^2, \\ \mu_{i+1}^{\mathrm{A}} &= \mu_i^{\mathrm{A}} - \langle x^{\mathrm{S}}, s_{\mathcal{I}_{i+1}} \rangle, \;\; \mu_{i+1}^{\mathrm{F}} = \mu_i^{\mathrm{F}} + \langle x^{\mathrm{S}}, d_{\mathcal{I}_{i+1}} \rangle, \\ x_{i+1}^{\mathrm{A}} &= x_i^{\mathrm{A}} - s_{\mathcal{I}_{i+1}} \;\text{ and }\; d_{i+1}^{\mathrm{F}} = d_i^{\mathrm{F}} + d_{\mathcal{I}_{i+1}}, \end{aligned}$$

increment i by 1 and return to Step 1.

4. Find the next set of indices that change status: Let $\alpha_{i+1} = \beta^{-1} \alpha_i$, and compute

$$\mathcal{J}_{i+1} = \{j : \alpha_i < \alpha_j^{\mathrm{B}} \leq \alpha_{i+1}\}$$

using the Heapsort algorithm.

5. Update the components of the objective and its slope: Compute

$$p_{i+1} = A_{\mathcal{J}_{i+1}} s_{\mathcal{J}_{i+1}} \text{ and } q_{i+1} = A_{\mathcal{J}_{i+1}} d_{\mathcal{J}_{i+1}}$$

update

$$\begin{aligned}
f^{\mathrm{C}}_{i+1} &= f^{\mathrm{C}}_i + \langle r^{\mathrm{A}}_i, p_{i+1}\rangle + \tfrac{1}{2}\|p_{i+1}\|^2,\\
f^{\mathrm{L}}_{i+1} &= f^{\mathrm{L}}_i - \langle r^{\mathrm{A}}_i, q_{i+1}\rangle + \langle r^{\mathrm{F}}_i, p_{i+1}\rangle - \langle p_{i+1}, q_{i+1}\rangle,\\
f^{\mathrm{Q}}_{i+1} &= f^{\mathrm{Q}}_i - \langle r^{\mathrm{F}}_i, q_{i+1}\rangle + \tfrac{1}{2}\|q_{i+1}\|^2,\\
\gamma^{\mathrm{A}}_{i+1} &= \gamma^{\mathrm{A}}_i + \langle r^{\mathrm{S}}, p_{i+1}\rangle, \quad \gamma^{\mathrm{F}}_{i+1} = \gamma^{\mathrm{F}}_i - \langle r^{\mathrm{S}}, q_{i+1}\rangle,\\
r^{\mathrm{A}}_{i+1} &= r^{\mathrm{A}}_i + p_{i+1} \text{ and } r^{\mathrm{F}}_{i+1} = r^{\mathrm{F}}_i - q_{i+1},
\end{aligned}$$

as well as

$$\begin{aligned}
\rho^{\mathrm{C}}_{i+1} &= \rho^{\mathrm{C}}_i + \langle x^{\mathrm{A}}_i, s_{\mathcal{J}_{i+1}}\rangle + \tfrac{1}{2}\|p_{i+1}\|^2,\\
\rho^{\mathrm{L}}_{i+1} &= \rho^{\mathrm{L}}_i - \langle x^{\mathrm{A}}_i, d_{\mathcal{J}_{i+1}}\rangle + \langle d^{\mathrm{F}}_i, s_{\mathcal{J}_{i+1}}\rangle - \langle d_{\mathcal{J}_{i+1}}, s_{\mathcal{J}_{i+1}}\rangle,\\
\rho^{\mathrm{Q}}_{i+1} &= \rho^{\mathrm{Q}}_i - \langle d^{\mathrm{F}}_i, d_{\mathcal{J}_{i+1}}\rangle + \tfrac{1}{2}\|d_{\mathcal{J}_{i+1}}\|^2,\\
\mu^{\mathrm{A}}_{i+1} &= \mu^{\mathrm{A}}_i + \langle x^{\mathrm{S}}, s_{\mathcal{J}_{i+1}}\rangle, \quad \mu^{\mathrm{F}}_{i+1} = \mu^{\mathrm{F}}_i - \langle x^{\mathrm{S}}, d_{\mathcal{J}_{i+1}}\rangle,\\
x^{\mathrm{A}}_{i+1} &= x^{\mathrm{A}}_i + s_{\mathcal{J}_{i+1}} \text{ and } d^{\mathrm{F}}_{i+1} = d^{\mathrm{F}}_i - d_{\mathcal{J}_{i+1}}.
\end{aligned}$$

6. Check for an approximate extended arc minimizer: Compute

$$\begin{aligned}
f_{i+1} &= f^{\mathrm{C}}_{i+1} + \alpha_{i+1} f^{\mathrm{L}}_{i+1} + \alpha^2_{i+1} f^{\mathrm{Q}}_{i+1},\\
\rho_{i+1} &= \rho^{\mathrm{C}}_{i+1} + \alpha_{i+1} \rho^{\mathrm{L}}_{i+1} + \alpha^2_{i+1} \rho^{\mathrm{Q}}_{i+1},\\
\phi_{i+1} &= f_{i+1} + \sigma \rho_{i+1} \text{ and}\\
\gamma_{i+1} &= \gamma^{\mathrm{A}}_{i+1} + \alpha_{i+1} \gamma^{\mathrm{F}}_{i+1} + \sigma(\mu^{\mathrm{A}}_{i+1} + \alpha_{i+1} \mu^{\mathrm{F}}_{i+1}).
\end{aligned}$$

If $\phi_{i+1} > \phi(x^{\mathrm{S}}, \sigma) + \eta \gamma_{i+1}$ or $\alpha_{i+1} \geq \max_j \alpha^{\mathrm{B}}_j$, set

$$\alpha^{\mathrm{C}} = \alpha_i, \quad x^{\mathrm{C}} = P_{\mathcal{X}}[x^{\mathrm{S}} + \alpha^{\mathrm{C}} d], \quad f(x^{\mathrm{C}}) = f_i \text{ and } \phi(x^{\mathrm{C}}, \sigma) = \phi_i,$$

and stop. Otherwise, increment i by 1 and return to Step 4.

3.1 Regularized linear least-squares minimization with fixed variables

Algorithm 3.3: The preconditioned conjugate-gradient regularized-least-squares method

Given x_0, set $r_0 = Ax_0 - b$ and $g_0 = Z[A^T r_0 + \sigma x_0]$, and let $v_0 = Z[M^{-1} g_0]$ and $p_0 = -v_0$. For $k = 0, 1, \ldots$ until convergence, perform the iteration

$$\begin{aligned}
q_k &= Ap_k, \\
\alpha_k &= \langle g_k, v_k \rangle / (\langle q_k, q_k \rangle + \sigma \langle p_k, p_k \rangle), \\
x_{k+1} &= x_k + \alpha_k p_k, \\
r_{k+1} &= r_k + \alpha_k q_k, \\
g_{k+1} &= Z[A^T r_{k+1} + \sigma x_{k+1}], \\
v_{k+1} &= Z[M^{-1} g_{k+1}], \\
\beta_k &= \langle g_{k+1}, v_{k+1} \rangle / \langle g_k, v_k \rangle \;\text{ and} \\
p_{k+1} &= -v_{k+1} + \beta_k p_k.
\end{aligned}$$

Notice that throughout only components $[p_k]_i$, $i \in \mathcal{F}$, can be nonzero, and this should be exploited when forming Ap_k and $\langle p_k, p_k \rangle$. The preconditioner needs to take account of the regularization term, and, at the very least, $M = \text{diag}(A^T A) + \sigma I$ is appropriate.

Availability

The algorithms described have been implemented as the modern Fortran package `blls`, and the later is available as part of the GALAHAD library [4].