COBYQA Reference

Release 1.0.dev0

Tom M. Ragonneau

October 28, 2021

CONTENTS

Date October 28, 2021

This section presents in depth the derivative trust-region method employed by COBYQA to tackle nonlinear constrained problems.

CHAPTER ONE

OPTIMIZATION FRAMEWORK (`COBYQA`)

See also:

`minimize`, `OptimizeResult`

1.1 Statement of the problem

CHAPTER

TWO

LINEAR ALGEBRA (`COBYQA.LINALG`)

This section presents the mathematical frameworks underneath the subproblem solvers of COBYQA.

See also:

`bvcs`, `bvlag`, `bvtcg`, `cpqp`, `givens`, `lctcg`, `nnls`, `qr`

2.1 Geometry of the interpolation set

2.2 Bound constrained truncated conjugate gradient

When no linear nor nonlinear constraints are provided to COBYQA (that is when the considered problem is bound constrained), the trust-region subproblem to solve at each iteration is of the form

$$\begin{array}{ll} \min & f(x) = \langle x, g \rangle + \frac{1}{2}\langle x, Hx \rangle \\ \text{s.t.} & l \le x \le u, \\ & \|x\| \le \Delta, \\ & x \in \mathbb{R}^n, \end{array} \tag{2.1}$$

where $g \in \mathbb{R}^n$ approximates the gradient of the nonlinear objective function at the origin, $H \in \mathbb{R}^{n \times n}$ is a symmetric matrix that approximates the Hessian matrix of the nonlinear objective function at the origin, $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ are the lower and upper bounds of the problems (with $l < u$), $\Delta > 0$ is the current trust-region radius, and $\|\cdot\|$ is the Euclidean norm.

2.2.1 The unconstrained case

We assume in this section that the lower and upper bounds in (2.1) are respectively set to $-\infty$ and $+\infty$. Powell showed in [B1] that a trust-region method is convergent if the trust-region step $x^* \in \mathbb{R}^n$ satisfies

$$f(x^0) - f(x^*) \ge \gamma \|g\| \min\{\Delta, \|g\|/\|H\|\},$$

for some $\gamma > 0$, where $\|\cdot\|$ is the Euclidean norm. It is easy to see that a Cauchy step satisfies such a condition (with $\gamma = 1/2$). Therefore, to preserve the computational efficiency of a trust-region method, it is usual to solve inexactly problem (2.1) using the truncated conjugate gradient method of Steihaug [B3] and Toint [B4]. Given the initial values $x^0 = 0$ and $d^0 = -g$, it generates the sequence of iterates

$$\begin{cases} \alpha_k = -\langle d^k, g^k \rangle / \langle d^k, Hd^k \rangle, \\ \beta_k = \|g^{k+1}\|^2 / \|g^k\|^2, \\ x^{k+1} = x^k + \alpha_k d^k, \\ d^{k+1} = -g^k + \beta_k d^k, \end{cases}$$

where $g^k = \nabla f(x^k) = g + Hx^k$. The computations are stopped if either

1. $g^k = 0$ and $\langle d^k, Hd^k \rangle \geq 0$, in which case the global minimizer is found; or
2. $\|x^{k+1}\| \geq \Delta$ or $\langle d^k, Hd^k \rangle < 0$, in which case $x^k + \alpha_\Delta d^k$ is returned, where $\alpha_\Delta > 0$ is chosen so that $\|x^k + \alpha_\Delta d^k\| = \Delta$.

It is known that the truncated conjugate gradient method terminates after at most n iterations, and that the reduction in the objective function obtained with the truncated conjugate gradient method is at least half of the reduction obtained by the global minimizer.

2.2.2 The constrained case

We assume in this section that at least some values of the lower and upper bounds in (2.1) are finite. When no linear nor nonlinear constraints are provided, COBYQA solves the trust-region subproblem using the TRSBOX algorithm [B2], which is presented below.

The bound constrained truncated conjugate gradient procedure

The strategy employed by `bvtcg` to tackle the bound constraints in the truncated conjugate gradient procedure is the use of an active set. At each iteration of the method, a truncate conjugate gradient step is performed on the coordinates that are not fixed by the active set. If a new bound is hit during such iteration, the bound is added to the active set, and the procedure is restarted. The active set is only enlarged through the iterations, which then ensures the termination of the method.

The initial active set is a subset of the active bounds at the origin. Clearly, a active bound should not be included in the active set if a Cauchy step (a positive step along $-g$) would depart from the bound, as the bound is never removed from the active set. The complete framework of `bvtcg` is described below. For sake of clarity, we denote $\mathcal{I}$ the active set and $\Pi(v)$ the vector whose i-th coordinate is v_i if $i \notin \mathcal{I}$, and zero otherwise.

1. Set $x^0 = 0$ and the active set $\mathcal{I}$ to the indices for which either $l_i = 0$ and $g_i \geq 0$ or $u_i = 0$ and $g_i \leq 0$.
2. Set $d^0 = -\Pi(g)$, $g^0 = g$, and $k = 0$.
3. Let $\alpha_{\Delta,k}$ be the largest number such that $\|x^k + \alpha_{\Delta,k} d^k\| = \Delta$.
4. Let $\alpha_{Q,k}$ be $-\langle d^k, g^k \rangle / \langle d^k, Hd^k \rangle$ if $\langle d^k, Hd^k \rangle > 0$ and $+\infty$ otherwise.
5. Let $\alpha_{B,k}$ be the largest number such that $l \leq x^k + \alpha_{B,k} d^k \leq u$ and $\alpha_k = \min\{\alpha_{\Delta,k}, \alpha_{Q,k}, \alpha_{B,k}\}$.
6. Update $x^{k+1} = x^k + \alpha_k d^k$, $g^{k+1} = g^k + \alpha_k Hd^k$, and $\beta_k = \|g^{k+1}\|^2 / \|g^k\|^2$.
7. If $\alpha_k = \alpha_{\Delta,k}$ or $g^{k+1} = 0$, stop the computations.
8. If $\alpha_k = \alpha_{B,k}$, add a new active coordinate to $\mathcal{I}$, set $x^0 = x^{k+1}$, and go to step 2.
9. Update $d^{k+1} = -\Pi(g^k) + \beta_k d^k$, increment k, and go to step 3.

Further refinement of the trial step

If the step x^k returned by the constrained truncated conjugate gradient procedure satisfies $\|x^k\| = \Delta$, it is likely that the objective function in (2.1) can be further decreased by moving this point round the trust-region boundary. In fact, the global solution of problem (2.1) is on the trust-region boundary. The method `bvtcg` then may further reduce the function evaluation by returning in this case an approximate solution to

$$\begin{aligned}
\min \quad & f(x) = \langle x - x^0, g \rangle + \tfrac{1}{2} \langle x - x^0, H(x - x^0) \rangle \\
\text{s.t.} \quad & l \leq x \leq u, \\
& \|x - x^0\| = \Delta, \\
& x \in \operatorname{span}\{\Pi(x^k), \Pi(g^k)\} \subseteq \mathbb{R}^n.
\end{aligned}$$

To do so, the method builds an orthogonal basis $\{\Pi(x^k), s\}$ of $\operatorname{span}\{\Pi(x^k), \Pi(g^k)\}$ by selecting the vector $s \in \mathbb{R}^n$ such that $\langle s, \Pi(x^k)\rangle = 0$, $\langle s, \Pi(g^k)\rangle < 0$, and $\|s\| = \|\Pi(x^k)\|$. Further, the method considers the function $x(\theta) = x^k + (\cos\theta - 1)\Pi(x^k) + \sin\theta s$ with $0 \le \theta \le \pi/4$ and solves approximately

$$\begin{array}{ll} \min & f(x(\theta)) \\ \text{s.t.} & l \le x(\theta) \le u, \\ & 0 \le \theta \le \pi/4, \end{array}$$

the trust-region condition being automatically ensured by the choice of s. If the value of θ is restricted by a bound, it is added to the active set $\mathcal{I}$, and the refinement procedure is restarted. Since the active set is very reduced, this procedure terminates in at most $n - |\mathcal{I}|$, where $|\mathcal{I}|$ denotes the number of active bounds at the end of the constrained truncated conjugate gradient procedure.

2.3 Convex piecewise quadratic programming

In general (that is when some linear and/or nonlinear constraints are provided), to determine a trust-region normal step, COBYQA must solve a problem of the form

$$\begin{array}{ll} \min & f(x) = \frac{1}{2}\big(\|[Ax - b]_+\|^2 + \|Cx - d\|^2\big) \\ \text{s.t.} & l \le x \le u, \\ & \|x\| \le \Delta, \\ & x \in \mathbb{R}^n, \end{array} \tag{2.2}$$

where $A \in \mathbb{R}^{m_1 \times n}$, $b \in \mathbb{R}^{m_1}$, $C \in \mathbb{R}^{m_2 \times n}$, $d \in \mathbb{R}^{m_2}$, $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ are the lower and upper bounds of the problems (with $l < u$), $\Delta > 0$ is the current trust-region radius, and $\|\cdot\|$ is the Euclidean norm.

2.4 Linear constrained truncated conjugate gradient

In general (that is when some linear and/or nonlinear constraints are provided), to determine a trust-region tangential step, COBYQA must solve a problem of the form

$$\begin{array}{ll} \min & f(x) = \langle x, g\rangle + \frac{1}{2}\langle x, Hx\rangle \\ \text{s.t.} & Ax \le b, \\ & Cx = d, \\ & \|x\| \le \Delta, \\ & x \in \mathbb{R}^n, \end{array} \tag{2.3}$$

where $g \in \mathbb{R}^n$ approximates the gradient of the nonlinear objective function at the origin, $H \in \mathbb{R}^{n \times n}$ is a symmetric matrix that approximates the Hessian matrix of the nonlinear objective function at the origin, $A \in \mathbb{R}^{m_1 \times n}$ and $C \in \mathbb{R}^{m_2 \times n}$ are the Jacobian matrices of the linear inequality and equality constraints, $b \in \mathbb{R}^{m_1}$ and $d \in \mathbb{R}^{m_2}$ are the corresponding right-hand sides, $\Delta > 0$ is the current trust-region radius, and $\|\cdot\|$ is the Euclidean norm.

2.4.1 The unconstrained case

As mentioned in the section *The unconstrained case* of the description of the `bvtcg` method, it is usual to solve inexactly problem (2.3) using a truncated conjugate gradient method of Steihaug [L3] and Toint [L4] in the unconstrained case. Given the initial values $x^0 = 0$ and $d^0 = -g$, it generates the sequence of iterates

$$\begin{cases} \alpha_k = -\langle d^k, g^k\rangle / \langle d^k, Hd^k\rangle, \\ \beta_k = \|g^{k+1}\|^2 / \|g^k\|^2, \\ x^{k+1} = x^k + \alpha_k d^k, \\ d^{k+1} = -g^k + \beta_k d^k, \end{cases}$$

where $g^k = \nabla f(x^k) = g + Hx^k$. The computations are stopped if either

1. $g^k = 0$ and $\langle d^k, Hd^k \rangle \geq 0$, in which case the global minimizer is found; or
2. $\|x^{k+1}\| \geq \Delta$ or $\langle d^k, Hd^k \rangle < 0$, in which case $x^k + \alpha_\Delta d^k$ is returned, where $\alpha_\Delta > 0$ is chosen so that $\|x^k + \alpha_\Delta d^k\| = \Delta$.

2.4.2 The constrained case

When linear and/or nonlinear constraints are provided, COBYQA solves its trust-region tangential subproblems using a modified TRSTEP algorithm [L2]. It is an active-set variation of the truncated conjugate gradient algorithm, which maintains the QR factorization of the matrix whose columns are the gradients of the active constraints. As for `bvtcg`, if a new constraint is added to the active set, the procedure is restarted. However, we allow constraints to be removed from the active set in this method.

Management of the active set

For convenience, we denote a_j, for $j \in \{1, 2, \dots, m_1\}$ and c_j, for $j \in \{1, 2, \dots, m_2\}$ the rows of the matrices A and C. We assume that the initial guess x^0 is feasible and that an inequality constraint $b_j - \langle a_j, x \rangle$ is positive and tiny for some $j \leq m_1$. If j do not belong to the active set and if $\langle a_j, g \rangle < 0$, then it is likely that the $\|x^1 - x^0\|$ is small, as a step along the search direction $d^0 = -g$ quickly exits the feasible set. Therefore, we must consider a constraint active whenever its residual becomes small. More precisely, for some feasible $x \in \mathbb{R}^n$, we let

$$\mathcal{J}(x) = \{j \leq m_1 : b_j - \langle a_j, x \rangle \leq \eta \Delta \|a_j\|\},$$

where η is some positive constant (set to $\eta = 0.2$ in `lctcg`). At each iteration, the active set is a subset of $\mathcal{J}(x^k)$. Moreover, the initial search direction d^0 should be close to $-g$ and prevent the point x^1 to be close from x^0. Therefore, the initial search direction d^0 is the unique solution of

$$\begin{array}{ll} \min & \frac{1}{2}\|g + d\|^2 \\ \text{s.t.} & \langle a_j, d \rangle \leq 0,\ j \in \mathcal{J}(x^k), \\ & \langle c_j, d \rangle = 0,\ j \in \{1, 2, \dots, m_2\} \\ & d \in \mathbb{R}^n. \end{array}$$

If $\langle a_j, d^0 \rangle < 0$ for some j, then the point x^1 will be further from this constraint than the initial guess. Therefore, the active set $\mathcal{I}$ is chosen to be $\{j \in \mathcal{J}(x^0) : \langle a_j, d^0 \rangle = 0\}$ (or a subset of it, chosen so that $\{a_j : j \in \mathcal{I}\}$ is a basis of span$\{a_j : j \in \mathcal{J}(x^0),\ \langle a_j, d^0 \rangle = 0\}$. bibtex spohinx The solution of such a problem is calculated using the Goldfarb and Idnani method for quadratic programming [L1].

2.5 Nonnegative least squares

The update mechanism of the Lagrange multipliers in COBYQA is based on a constrained least-squares problem, where some variables must remain nonnegative (in order to satisfy some complementary slackness conditions). The problem we solve is of the form

$$\begin{array}{ll} \min & f(x) = \frac{1}{2}\|Ax - b\|^2 \\ \text{s.t.} & x_i \geq 0,\ i = 1, 2, \dots, n_0, \\ & x \in \mathbb{R}^n, \end{array} \tag{2.4}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, n_0 is a nonnegative integer with $n_0 \leq n$, and $\|\cdot\|$ is the Euclidean norm. We observe that if $n_0 = 0$, problem (2.4) is a simple unconstrained least-squares problem, which can be solved using traditional methods (see, e.g., `numpy.linalg.lstsq`).

2.5.1 Description of the method

In order to solve problem (2.4) when $n_0 \geq 1$, we construct an active-set method based on Algorithm 23.10 of [N1], referred to as `nnls`. The framework of the method is described below.

1. Set the active set $\mathcal{I}^0$ initially to $\{1, 2, \dots, n_0\}$, the initial guess x^0 to the origin, and $k = 0$.
2. Evaluate the gradient of the objective function of (2.4) at x^k, namely $\nabla f(x^k) = A^\mathsf{T}(Ax^k - b)$.
3. If the KKT conditions for problem (2.4) hold at x^k, stop the computations.
4. Remove from the active set $\mathcal{I}^k$ an index yielding $\min\{\partial_i f(x) : i \in \mathcal{I}^k\}$ to build $\mathcal{I}^{k+1}$.
5. Let $A_{\mathcal{I}}^{k+1}$ be the matrix whose i-th column if the i-th column of A if $i \notin \mathcal{I}^{k+1}$, and zero otherwise.
6. Let z^{k+1} be a solution of the least squares $\min\|A_{\mathcal{I}}^{k+1} z - b\|$ with $z_i^{k+1} = 0$ for $i \in \mathcal{I}^{k+1}$, and increment k.
7. If $z_i^k > 0$ for all $i \notin \mathcal{I}^k$ with $i \leq n_0$, update $x^k = z^k$ and go to step 2. Set otherwise $x^k = x^{k-1}$.
8. Set $\alpha_k = \min\{x_i^k/(x_i^k - z_i^k) : z_i^k \leq 0,\ i \notin \mathcal{I}^k,\ i \leq n_0\}$.
9. Update $x^{k+1} = x^k + \alpha_k(z^k - x^k), \mathcal{I}^{k+1} = \mathcal{I}^k \cup \{i \leq n_0 : x_i^{k+1} = 0\}$, increment k, and go to step 5.

The unconstrained least-squares subproblem of `nnls` at step 6 is solved using `numpy.linalg.lstsq`. Several refinements of this framework have been made in the implementation.

1. The number of past iterations is maintained to stop the computations if a given threshold is exceeded.
2. Numerical difficulties may arise at step 8 of the method whenever the denominator comes close to zero (although it remains theoretically nonzero). The division is then safeguarded.
3. Computer rounding errors may engender infinite cycling if the solution has been found when checking the KKT conditions. Therefore, the computations are stopped if no progress is made from an iteration to another in terms of objective function reduction. The theoretical analysis below show that strict function decrease must occur from an iteration to another.

2.5.2 Convergence of the method

To study the theoretical properties of the framework, we regard it as consisting of two nested loops. The inner loop (steps 5–9) has unique entry and exit points at steps 5 and 7, and only broadens the active set. The outer loop (steps 2–9) also has unique entry and exit points at steps 2 and 3, and only narrows the active set. Since the termination criteria of the outer loop are the KKT conditions for problem (2.4), it is clear that the termination of the algorithm implies its convergence. In order to prove the termination of the algorithm, we necessitate the following result whose proof is established in Lemma 23.17 of [N1].

Lemma 1. Assume that a matrix $A \in \mathbb{R}^{m \times n}$ is a full column rank matrix (with $n \leq m$) and that a vector $b \in \mathbb{R}^m$ satisfies $\langle b, Ae_i \rangle = 0$ for $i \in \{1, 2, \dots, n\} \setminus \{j\}$ and $\langle b, Ae_j \rangle > 0$ with $j \in \{1, 2, \dots, n\}$. Then the solution vector x^* of the least-squares problem $\min\|Ax - b\|$ satisfies $x_j^* > 0$.

Assume that the KKT conditions for problem (2.4) do not hold at the origin. At the first iteration, the algorithm selects the index (j, say) of the most negative component of $\nabla f(0) = -A^\mathsf{T} b$ and remove it from the active set $\mathcal{I}^0$. According to Lemma 1, the solution of the least-squares problem at step 6 satisfies $z_j^1 > 0$. It is then easy to see that at each iteration, the solution satisfies $z_j^{k+1} > 0$, where j is the index selected at step 4. Such a solution is then modified by the inner loop to ensure that $x_i^{k+1} \geq 0$ for any $i \in \{1, 2, \dots, n_0\}$. To do so, it will select the closest point to z^{k+1} on the line joining x^k to z^{k+1} that is feasible.

Termination of the inner loop

It is clear at this step that all operations made in the inner loop are well-defined. Moreover, at each inner loop iteration, the cardinal number of the active set $\mathcal{I}^k$ is incremented. If $\mathcal{I}^k$ is maximal, that is $\mathcal{I}^k = \{1, 2, \dots, n_0\}$, then the condition at step 7 is always true, and the loop terminates. Therefore, the inner loop must terminate in at most $n_0 - |\mathcal{I}^k| + 1$ iterations, where $|\mathcal{I}^k|$ denotes the cardinal number of the active set when the first inner iteration started.

Termination of the outer loop

Termination of the outer loop can be directly inferred by showing that the value of f strictly decreases at each outer loop iteration. Such a condition ensures that the active set $\mathcal{I}^k$ at a given outer loop iteration is different from all its previous instances, as the feasibility of each iterate of the outer loop has already been shown (it is specifically the purpose of the inner loop).

When an inner loop iteration finishes, the value of x^{k+1} is solves

$$\begin{aligned} \min \quad & f(x) = \frac{1}{2}\|Ax - b\|^2 \\ \text{s.t.} \quad & x_i = 0, \ i \in \mathcal{I}^{k+1}, \\ & x_i > 0, \ i \notin \mathcal{I}^{k+1}, \ i \le n_0, \\ & x \in \mathbb{R}^n. \end{aligned}$$

If the KKT conditions for problem (2.4) hold at x^{k+1}, then termination occurs. Otherwise, after incrementing k, the index yielding the least value of $\nabla f(x^k)$ is removed from the active set $\mathcal{I}^k$, and the tentative solution vector z^{k+1} clearly provides $f(z^{k+1}) < f(x^k)$. The result is then proven if no inner loop is entertained (that is, if condition at step 7 holds). Otherwise, at the end of each inner loop, we have

$$\begin{aligned} \sqrt{2f(x^{k+1})} &= \big\|A(x^k + \alpha_k(z^k - x^k)) - b\big\| \\ &= \big\|(1 - \alpha_k)(Ax^k - b) + \alpha_k(Az^k - b)\big\| \\ &< \|Ax^k - b\| = \sqrt{2f(x^k)}, \end{aligned}$$

since $\alpha_k \in (0, 1)$. The termination of the outer loop is then proven, as well as the convergence of the method.