MATH-329 Nonlinear optimization
Exercise session 6: Truncated conjugate gradient for trust-region

Instructor: Nicolas Boumal
TAs: Guifré Sánchez, Antoine Gonon

Document compiled on October 13, 2025

1. Output of tCG in case of early termination. Consider the definition $v_n \triangleq v_{n-1} + tp_{n-1}$ and the equation $\|v_n\|^2 = \Delta^2$ (where the unknown is t) from the truncated conjugate gradients algorithm in the course.

1. Show that the equation is indeed quadratic in t.
2. In the context above, argue that there is one positive root and one negative root (in particular, they are distinct). Give an expression for the positive root.

It is possible to show that choosing the positive root induces the smaller model value.

Answer.

1. The quadratic equation is

$$t^2\|p_{n-1}\|^2 + 2t\,\langle p_{n-1}, v_{n-1}\rangle + \|v_{n-1}\|^2 - \Delta^2.$$

since $\|v_{n-1}\| < \Delta$ (otherwise, the algorithm would have stopped at iteration $n-1$), the equation discriminant is positive and there are two disctinct roots given by

$$\frac{-\langle p_{n-1}, v_{n-1}\rangle \pm \sqrt{\langle p_{n-1}, v_{n-1}\rangle^2 + \|p_{n-1}\|^2\,(\Delta^2 - \|v_{n-1}\|^2)}}{\|p_{n-1}\|^2}.$$

2. The positive root can be computed by choosing the + sign in the equation above.
3. We choose t to be positive so that we move in the direction of p_{n-1}. This choice guarantees that the model decreases; see Eqn. (6.19) of lecture notes.

   If the root is double, then $\|p_{n-1}\|^2\,(\Delta^2 - \|v_{n-1}\|^2)$. Either $p_{n-1} = 0$, which means that the algorithm has reached the standard termination criterion of CG, or $\|v_{n-1}\| = \Delta$, and we stopped at iteration $n-1$. Either case, it is fine to stop.

■

2. First iterate of tCG.

1. Verify that the first iterate of tCG, that is, v_1 as produced by the algorithm in lecture notes, is exactly the Cauchy step.

2. What can you conclude regarding the global convergence of the trust-region method with tCG?

Answer.

1. We consider three different scenarios for the first step of tCG.

   (a) $\langle p_0, Hp_0\rangle \leq 0$: then $v_1 = \frac{\Delta}{\|b\|}b$.

   (b) $\langle p_0, Hp_0\rangle > 0$ and $\|v_0^+\| \geq \Delta$ (meaning $\frac{\|b\|^3}{\langle b, Hb\rangle} \geq \Delta$): then $v_1 = \frac{\Delta}{\|b\|}b$.

   (c) $\langle p_0, Hp_0\rangle > 0$ and $\|v_0^+\| < \Delta$: we have $v_1 = \frac{\|b\|^2}{\langle b, Hb\rangle}$.

   Cases (b) and (c) can be written compactly as

   $$v_1 = \min\left(\frac{\|b\|^2}{\langle b, Hb\rangle}, \frac{\Delta}{\|b\|}\right) b,$$

   This matches the expression of the Cauchy step.

2. Since the model value decreases during tCG, its value at the last iterate v_N cannot be worse than that of the first iterate v_1, which corresponds to the Cauchy step. This means that the step u_k computed by tCG at iteration k of trust-region does at least as good as the Cauchy step, which is sufficient to guarantee global convergence (Section 6.2 of lecture notes).

■

3. Improving the TR method with tCG.

1. Implement the truncated conjugate gradients method (tCG) as explained in the lecture notes.

2. Implement the trust-region algorithm using the tCG algorithm to approximate the solution to the subproblem. You can simply modify the implementation you did for exercise session 5.

3. Run the trust-region algorithm with tCG on the multidimensional Rosenbrock function (see the definition at the end of the exercise sheet). We provide files on Moodle to compute the function value, the gradient and the Hessian. You may use $n = 10$, $x_0 =$ `randn(n,1)`.

4. Compare its performance on this problem with trust-region using the Cauchy point as approximate solution to the trust-region problem. Comment on your observations.

Answer.

1. See lecture notes for pseudo code.

2. See lecture notes for pseudo code.

3. The TR algorithm with tCG finds a point where the gradient norm is less than 10^{-10} in approximately 50 iterations. Due to random initialization, sometimes this critical point is the local minimum $(-1, 1, \ldots, 1)^\top$ and sometimes the global minimum $(1, \ldots, 1)^\top$. In Figure 1 we can appreciate the quadratic convergence of the functions values and gradient norms.

4. In previous exercise sessions we observed that TR with Cauchy point would need several dozens of thousands of iterations to reach an approximate critical point (see Figure 2). The tremendous improvement we observe when using tCG suggests how suboptimal the Cauchy point is as a solution to the TR subproblem. Yet, the Cauchy point is sufficient to guarantee global convergence: it remains an acceptable worst case alternative when the approximate solution to the TR subproblem is not satisfactory (with tCG we always get something at least as good but other strategies to solve the subproblem might not).

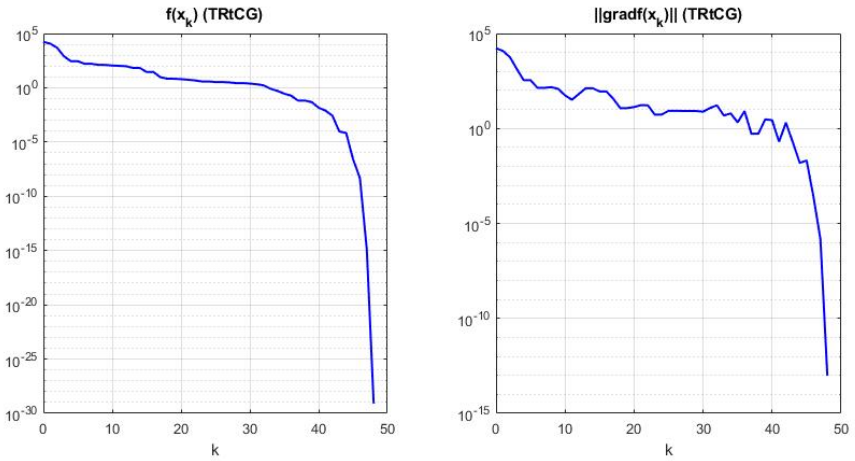

Figure 1: Objective value and gradient norm throughout the iterations of TR with tCG on the multidimensional Rosenbrock function ($n = 10$, $x_0 =$ `randn(n,1)`).

■

Multidimensional Rosenbrock function. We generalize the Rosenbrock function in n dimensions as

$$f(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (1 - x_i)^2 \right].$$

The vector of ones is the unique global minimum (because the function is non-negative and is zero if and only if all entries are ones). The gradient at $x \in \mathbb{R}^n$ is given by

$$\nabla f(x)_i = \begin{cases} -2(1 - x_1) - 400x_1(x_2 - x_1^2) & \text{if } i = 1 \\ 200(x_i - x_{i-1}^2) - 2(1 - x_i) - 400x_i(x_{i+1} - x_i^2) & \text{if } 1 < i < n \\ 200(x_n - x_{n-1}^2) & \text{if } i = n. \end{cases}$$

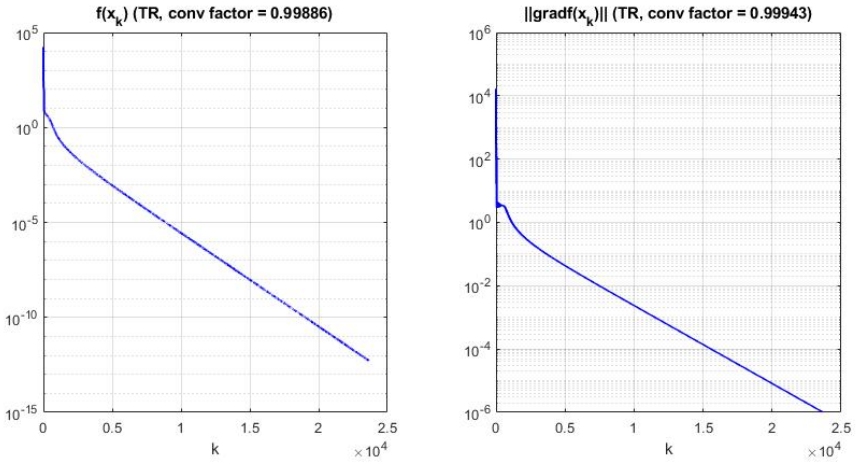

Figure 2: Objective value and gradient norm throughout the iterations of TR with Cauchy point on the multidimensional Rosenbrock function ($n = 10$, $x_0 =$ `randn(n,1)`).

The Hessian at x is a symmetric tridiagonal $n \times n$ matrix. The main diagonal and the first diagonal above are given by

$$\begin{bmatrix} 2 + 1200x_1^2 - 400x_2 \\ 202 + 1200x_2^2 - 400x_3 \\ \vdots \\ 202 + 1200x_{n-1}^2 - 400x_n \\ 200 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -400x_1 \\ \vdots \\ -400x_{n-1} \end{bmatrix}$$

respectively. In practice we never build the full matrix but solely compute matrix/vector products. This can be done efficiently because the matrix is sparse.