Trust-Region Method

Yuanshen Lin

January 6, 2024

Abstract

In the realm of numerical optimization, Trust Region methods stand out as a robust and efficient class of algorithms for solving nonlinear optimization problems. Trust Region methods determine both the step length and the direction by solving a subproblem within a region around the current iterate. This region, or "trust region", is adapted at each iteration based on the local behavior of the objective function.

This report delves into the fundamentals of Trust-Region methods, beginning with its idea and algorithm.Taking second-order approximation as an example of subproblem, we consider the way to solve it. Furthermore, we analysis the convergence of the method and find it is superlinear. Lastly, we compare the performance of trust-region with other algorithms on the Rosenbrock function.

Kewords: Trust Region Methods, Nonlinear Optimization, Numerical Algorithms, Convergence Analysis.

1 Introduction

In the traditional line search methods, we generate a search direction and then find a suitable step length. There is a limitation that we cannot change it into a better search direction such that maybe objective function decreases slowly. To search for a better direction and step length, trust-region methods define a region around the current iterate within which they trust the model to be an adequate representation of the objective function and choose a step to be the approximate minimizer of the model in this region. In general, the direction of the step changes whenever the size of the trust region is altered. Trust-region is an improvement of traditional line search methods. It combines the idea of nonlinear programming and split the problem into many subproblems.

The advantages of trust-region method:

- Robust Convergence: Trust region methods are known for their robust convergence properties, especially when dealing with difficult optimization problems.
- Adaptive Steps: They adaptively adjust the step size based on the quality of the approximation, leading to efficient and reliable optimization steps.
- Global Convergence: Trust region methods have been shown to have globally convergent modifications, making them suitable for a wide range of optimization issues.

There are also some disadvantages of trust-region methods:

- Complexity: They can be more complex to implement compared to other methods like line-search, due to the need for solving nonlinear subproblems.

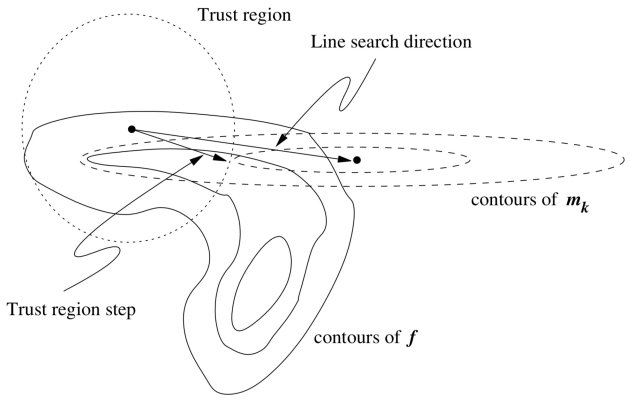

Figure 1: Trust-region and line search steps

- Computation Cost: The methods may require solving subproblems that are more computationally intensive, especially when dealing with large-scale problems.
- Parameter Tuning: Determining the appropriate size of the trust region and adjusting it can be challenging and may require careful tuning.

2 Idea of Trust-Region Method

In this essay, we consider the model function m_k as the second-order Taylor series approximation of the objective function at each iterate x_k. Assume the objective function f is twice continuously differentiable, then the model function m_k is given by

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k) p \approx f(x_k + p) \tag{1}$$

It is a precise approximation of f at x_k if p is small as the error is $O(||p||^3)$.

So, we need to give a good approximation of f with a small p. We define a trust region $B_{\Delta_k} x_k$ and require that trust-region radius Δ_k is small to ensure that the approximation is precise. Then we can solve the subproblem:

$$\begin{aligned} &\min_{p \in \mathbb{R}^n} m_k(p) = f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T \nabla^2 f(x_k) p \\ &\text{s.t.} \quad ||p|| \leq \Delta_k \end{aligned} \tag{2}$$

In the subproblem, the Hessian matrix and even the objective function can be replaced with other good approximations. For simplicity, we only consider the case above in the following analysis.

As we claim in the introduction, the trust-region method is sensitive to the choice of trust-region radius Δ_k at each iteration. So we develop a ratio ρ_k to evaluate the radius and adjust it in time.

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \tag{3}$$

where p_k is the minimizer of the subproblem.According to ρ_k, we can decide whether to accept p_k. If p_k is good enough, the trust-region radius Δ_k can become larger. If not, we minify Δ_k.

$$x_{k+1} = \begin{cases} x_k, & if \ \ \rho_k \leq p_0 \\ x_k + p_k, & otherwise \end{cases} \tag{4}$$

$$\Delta_{k+1} = \begin{cases} c_0\Delta_k, & if \ \ \rho_k \leq p_1 \\ \Delta_k, & if \ \ p_1 < \rho_k < p_2 \\ \min\{c_1\Delta_k, \Delta_{\max}\}, & otherwise \end{cases} \tag{5}$$

where $0 \leq p_0 < p_1 < p_2 < 1$, $0 < c_0 < 1 < c_1$, $\Delta_{\max} > 0$ are constant. In pratice, we usually take $p_0 = 0.001, p_1 = 0.25, p_2 = 0.75, c_0 = 0.25, c_1 = 2$.

The following algorithm1 is the trust-region method we have discussed above.

Algorithm 1: Trust-Region Method

Input: Initial point x_0, initial trust-region radius Δ_0, tolerance ϵ
Output: Optimal point x^*

while $||\nabla f(x_k)|| > \epsilon$ **do**
  solve the subproblem(6) to get p_k
  compute ρ_k as(7)
  if $\rho_k \leq p_0$ **then**
    $\Delta_{k+1} = c_0\Delta_k$
  else
    if $p_1 < \rho_k < p_2$ **then**
      $\Delta_{k+1} = \Delta_k$
    else
      $\Delta_{k+1} = \min\{c_1\Delta_k, \Delta_{\max}\}$
    end
  end
  if $\rho_k \leq p_0$ **then**
    $x_{k+1} = x_k$
  else
    $x_{k+1} = x_k + p_k$
  end
end
return $x^* = x_k$

3 Algorithm for Subproblems

Theorem 3.1. *The vector p^* is a global solution of the trust-region problem*

$$\begin{aligned} &\min_{p\in\mathbb{R}^n} m_k(p) = f(x_k) + g^T p + \frac{1}{2}p^T Bp \\ &s.t. \quad ||p|| \leq \Delta_k \end{aligned} \tag{6}$$

if and only if p^* is feasible and there is a scalar $\lambda \geq 0$ such that the following conditions are satisfied:

$$\begin{aligned} (B + \lambda I)p^* &= -g \\ \lambda(\Delta - ||p^*||) &= 0 \\ (B + \lambda I) \text{ is positive semidefinite} & \end{aligned} \tag{7}$$

Proof. **To prove sufficiency**, we create the Lagrangian function:

$$L(p, \lambda) = f + g^T p + \frac{1}{2} p^T B p - \frac{\lambda}{2}(\Delta^2 - ||p||^2)$$

where $\lambda \geq 0$. By KKT conditions, p^* is the optimal, we will have

$$\begin{aligned} (B + \lambda I)p^* + g &= 0 \\ \frac{\lambda}{2}(\Delta^2 - ||p^*||^2) &= 0 \end{aligned}$$

So we get

$$(B + \lambda I)p^* = -g \tag{8}$$

$$\lambda(\Delta - ||p^*||) = 0 \tag{9}$$

To get the last condition, $\forall p, ||p|| = \Delta$

$$m(p) \geq m(p^*) = m(p^*) + \frac{\lambda}{2}(||p^*||^2 - ||p||^2)$$

Estimate g by the equality we get above, we get the last condition:

$$(p - p^*)^T (B + \lambda I)(p - p^*) \geq 0$$

which means $(B + \lambda I)$ is positive semidefinite.

To prove necessity, we create a function:

$$\hat{m}(p) = m(p) + \frac{\lambda}{2} p^T p$$

According to the three conditions, $\hat{m}(p)$ is convex and $\nabla \hat{m}(p^*) = 0$. We can tell that p^* is the minimizer of $\hat{m}(p)$. For all feasible p,

$$\hat{m}(p) \geq \hat{m}(p^*) \Rightarrow m(p) \geq m(p^*) + \frac{\lambda}{2}(||p^*||^2 - ||p||^2) \geq m(p^*)$$

□

To compute this p^*, we define the following function:

$$p(\lambda) = -(B + \lambda I)^{-1} g$$

Now we need to find suitable λ such that7 holds.

Suppose $B = Q \Lambda Q^T$, where $Q = [q_1, q_2, \ldots, q_n]$ is orthogonal, $\Lambda = diag(\lambda_1, \ldots, \lambda_n)$, $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. Only consider $\lambda_1 \leq 0$.For $\lambda > -\lambda_1 \geq 0$,

$$p(\lambda) = -Q(\Lambda + \lambda I)^{-1} Q^T g = -\sum_{i=1}^{n} \frac{q_i^T g}{\lambda_i + \lambda} q_i$$

$$||p(\lambda)||^2 = \sum_{i=1}^{n} \frac{\left(q_i^T g\right)^2}{\left(\lambda_i + \lambda\right)^2}$$

It is a strictly decreasing function with

$$\lim_{\lambda \to \infty} ||p(\lambda)|| = 0, \qquad \lim_{\lambda \to \lambda_1^+} ||p(\lambda)|| = \infty$$

So the solution of $||p(\lambda)|| = \Delta$ is existing and unique. We can use some numerical method like **Newton method** to compute the λ^* and p^*

Besides, there are some way like **Dogled method, Two-Dimensional Subspace Minimization and Steihaug-CG** to approach the minimizer of the subproblem. They also work precisely on it.

4 Convergence Analysis

Definition 4.1. *Suppose $m_k(p)$ is the second-order Taylor series approximation of f at $x = x^k$, the constant τ_k is the optimal of the problem:*

$$\begin{aligned} &\min m_k(-\tau \nabla f(x_k)) \\ &s.t. ||\tau \nabla f(x_k)|| \leq \Delta_k, \tau \geq 0 \end{aligned}$$

*The **Cauchy point** is $x_k^C = x_k - \tau_k \nabla f(x_k)$, and $p_k^C = -\tau_k \nabla f(x_k)$ is called **Cauchy step**.*

It is easy to get

$$p_k^C = -\tau_k \frac{\Delta_k}{||g_k||} g_k \tag{10}$$

where

$$\tau_k = \begin{cases} 1, & if \quad g_k^T B_k g_k \leq 0 \\ \min(\frac{||g_k||^3}{\Delta_k g_k^T B_k g_k}, 1), & \text{otherwise} \end{cases} \tag{11}$$

Actually, it is a gredient decent with precise line search method. In practice, we do not use it to approach next iteration without use the message of B_k sufficiently. But when we analysis the convergence, Cauchy point contains the convergence though it is weaker than real iteration.

Lemmma 4.1. *The Cauchy point p_k^c satisfies the following inequality with $p_1 = \frac{1}{2}$:*

$$m_k(0) - m_k(p_k^c) \geq \frac{1}{2} ||g_k|| \min(\Delta_k, \frac{||g_k||}{||B_k||}) \tag{12}$$

Proof. For simplicity, we drop the iteration index k in the proof.

When $g^T B g \leq 0$,

$$\begin{aligned} m(p^c) - m(0) &= m(-\frac{\Delta g}{||g||}) - f \\ &= -\Delta ||g|| + \frac{1}{2} \frac{\Delta^2}{||g||^2} g^T B g \\ &\leq -\Delta ||g|| \end{aligned}$$

When $g^T Bg > 0$ and $\frac{||g||^3}{\Delta g^T Bg} \leq 1$.

$$\begin{aligned} m(p^c) - m(0) &= -\frac{||g||^4}{g^T Bg} + \frac{1}{2} g^T Bg \frac{||g||^4}{(g^T Bg)^2} \\ &= -\frac{1}{2} \frac{||g||^4}{g^T Bg} \\ &\leq -\frac{1}{2} \frac{||g||^2}{||B||} \end{aligned}$$

In the remaining case, $g^T Bg < \frac{||g||^3}{\Delta}$:

$$\begin{aligned} m(p^c) - m(0) &= -\Delta ||g|| + \frac{1}{2} \frac{\Delta g^T Bg}{||g||^2} \\ &\leq -\frac{1}{2} \Delta ||g|| \end{aligned}$$

In conclusion, (12) holds.

□

4.1 Global Convergence

Theorem 4.2. *Let $p_0 = 0$. Suppose that $||B_k|| \leq \beta$ for some constant β, that f is bounded below o on the level set $S = \{x | f(x) \leq f(x_0)\}$ and Lipschitz continuously differentiable in the neighborhood $S(R_0) = \{x | ||x - y|| < R_0$ for $y \in S\}$ for some $R_0 > 0$, and that all approximate solutions of we then have*

$$\liminf_{k \to \infty} ||g_k|| = 0 \tag{13}$$

Proof.

$$\begin{aligned} |\rho_k - 1| &= |\frac{f(x_k) - f(x_k + p_k) - m_k(0) + m_k(p_k)}{m_k(0) - m_k(p_k)}| \\ &= |\frac{m_k(p_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}| \end{aligned}$$

Using Taylor's theorem, β_1 is the Lipschitz constant.

$$\begin{aligned} |m_k(p_k) - f(x_k + p_k)| &= |\frac{1}{2} p_k^T B_k p_k - \int_0^1 [g(x_k + t p_k) - g(x_k)]^T p_k dt| \\ &\leq \frac{\beta}{2} ||p_k||^2 + \beta_1 ||p_k||^2 \end{aligned} \tag{14}$$

Suppose for contradiction that there is $\epsilon > 0$ and a positive index K such that

$$||g_k|| \geq \epsilon \quad \forall k \geq K \tag{15}$$

By the inequality above, we can have

$$|\rho_k - 1| \leq \frac{\gamma^2 \Delta_k^2 (\frac{\beta}{2} + \beta_1)}{c_1 \epsilon \min(\Delta_k, \frac{\epsilon}{\beta})} \tag{16}$$

Give a threshold $\bar{\Delta}$

$$\bar{\Delta} = \min(\frac{c_1\epsilon}{\gamma^2(\beta + 2\beta_1)}, \frac{R_0}{\gamma})$$

If $\Delta_k \leq \bar{\Delta}$, it can cause:

$$|\rho_k - 1| \leq \frac{\gamma^2\Delta_k(\frac{\beta}{2} + \beta_1)}{c_1\epsilon} \leq \frac{\gamma^2\bar{\Delta}(\frac{\beta}{2} + \beta_1)}{c_1\epsilon} \leq \frac{1}{2}$$

$\rho_k > \frac{1}{4}$. According to the Algorithm, we have $\Delta_{k+1} \geq \Delta_k$ such that Δ_k is increasing until larger than $\bar{\Delta}$.

then we conclude that $\Delta_k \geq \min(\Delta_K, \frac{\bar{\Delta}}{4})$

But consider that:

$$\begin{aligned} f(x_k) - f(x_{k+1}) &= f(x_k) - f(x_k + p_k) \\ &\geq \frac{1}{4}[m_k(0) - m_k(p_k)] \\ &\geq \frac{1}{4}c_1\epsilon \min(\Delta_k, \frac{\epsilon}{\beta}) \end{aligned}$$

Since f is bounded below, it follows from this inequality that

$$\lim_{k\to\infty} \Delta_k = 0$$

which is contradicting to the inequality above. So we must have $\rho_k < \frac{1}{4}$ for k sufficiently large. In this case, Δ_k will eventually be multiplied by $\frac{1}{4}$ at every iteration, then we have $\lim_{k\to\infty} \Delta_k = 0$, which again contradicts. Hence,

$$\liminf_{k\to\infty} ||g_k|| = 0$$

□

Theorem 4.3. *Let $p_0 \in (0, \frac{1}{4})$.Suppose that $||B_k|| \leq \beta$ for some constant β, that f is bounded below o on the level set $S = \{x|f(x) \leq f(x_0)\}$ and Lipschitz continuously differentiable in the neighborhood $S(R_0) = \{x|||x-y|| < R_0$ for $y \in S\}$ for some $R_0 > 0$.And p_k satisfy the inequality 12. Then we have*

$$\lim_{k\to\infty} g_k = 0 \tag{17}$$

4.2 Local Convergence

Theorem 4.4. *Let f is twice Lipschitz continuously differentiable in a neighborhood of a x^* at which second-order sufficient conditions are satisfied. Supposed the sequence $\{x_k\}$ converges to x^* and that for all k sufficiently large, $B_k = \nabla^2 f(x_k)$ and p_k is chosen as Cauchy point satisfying 12 and for Newton step p_k^N whenever $||p_k^N|| \leq \frac{1}{2}\Delta_k$, $||p_k - p_k^N|| = o(||p_k^N||)$.*

*Then the trust-region bound Δ_k becomes inactive for all k sufficiently large and the sequence $\{x_k\}$ converges **superlinearly** to x^**

Proof. Following the assumption above, we have

$$||p_k|| \leq \Delta_k < 2||p_k^N|| \leq 2||\nabla^2 f(x_k)^{-1}||||g_k||$$

So we get

$$||g_k|| \geq \frac{||p_k||}{2||\nabla^2 f(x_k)^{-1}||}$$

Then we have

$$\begin{aligned} m_k(0) - m_k(p_k) &\geq c_1 ||g_k|| \min(\Delta_k, \frac{||g_k||}{||B_k||}) \\ &\geq c_1 \frac{||p_k||^2}{4||\nabla^2 f(x_k)^{-1}||^2 ||\nabla^2 f(x_k)||} \end{aligned}$$

□

Because $x_k \to x^*$, we use the continuity of $\nabla^2 f(x)$ to loose the parameter

$$c_3 = \frac{c_1}{8||\nabla^2 f(x^*)^{-1}||^2 ||\nabla^2 f(x^*)||} \leq \frac{c_1}{4||\nabla^2 f(x_k)^{-1}||^2 ||\nabla^2 f(x_k)||}$$

Then we have $m_k(0) - m_k(p_k) \geq c_3 ||p_k||^2$, $c_3 > 0$ By Taylor's theorem, we have

$$\begin{aligned} &|f(x_k) - f(x_k + p_k) - m_k(0) + m_k(p_k)| \\ &= |\frac{1}{2} p_k^T \nabla^2 f(x_k) p_k - \frac{1}{2} \int_0^1 p_k^T \nabla^2 f(x_k + t p_k) p_k dt| \\ &\leq \frac{L}{4} ||p_k||^3 \end{aligned}$$

where $L > 0$ is the Lipschitz constant for the Hessian. For sufficiently large k,

$$|\rho_k - 1| \leq \frac{||p_k||^3 L}{4c_3 ||p_k||^2} = \frac{L||p_k||}{4c_3} \leq \frac{L}{4c_3} \Delta_k$$

Δ_k can be reduced only if $\rho_k < \frac{1}{4}$. By the inequality above, if Δ_k is small, $\rho_k > \frac{1}{4}$ and Δ_k cannot be reduced. So $\{\Delta_k\}$ is bounded away from 0. Since $x_k \to x^*$, we have $||p_k^N|| \to 0$ and $||p_k|| \to 0$. So $||p_k^N|| \leq \frac{1}{2}\Delta_k$ is always satisfied.

Following the convergence of Newton's method, we have

$$||x_k + p_k^N - x^*|| = o(||x_k - x^*||^2)$$

Then we can eventually get

$$\begin{aligned} &||x_k + p_k - x^*|| \\ &\leq ||x_k + p_k^N - x^*|| + ||p_k^N - p_k|| \\ &= o(||x_k - x^*||^2) + o(||p_k^N||) \\ &= o(||x_k - x^*||) \end{aligned}$$

Thus proving superlinearly convergence.

5 Numerical Experiment

Consider Rosenbrock Function.

In mathematical optimization, the Rosenbrock function is a non-convex function, which is used as a performance test problem for optimization algorithms.

The global minimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial. To converge to the global minimum, however, is difficult.

$$f(x) = \sum_{i=1}^{\frac{N}{2}} [100(x_{2i-1}^2 - x_{2i})^2 + (x_{2i-1} - 1)^2]$$

5.1 2D case

$$f(x, y) = 100(y - x^2)^2 + (1 - x^2)$$

It has only one minimizer $(1, 1)$.

Compare the Trust-Region with Nelder-Mead method and Newton-CG. We take the start point $x_0 = (1.3, 0)$.

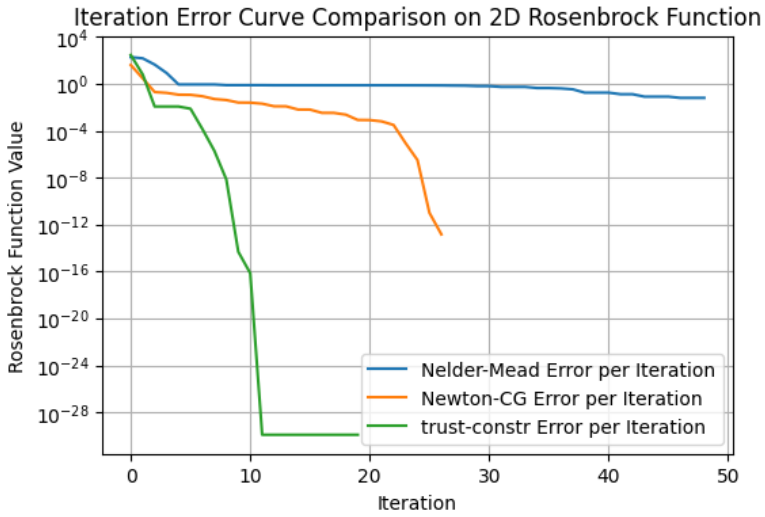

Figure 2: Iteration error of three algorithms(y-axis is log scale)

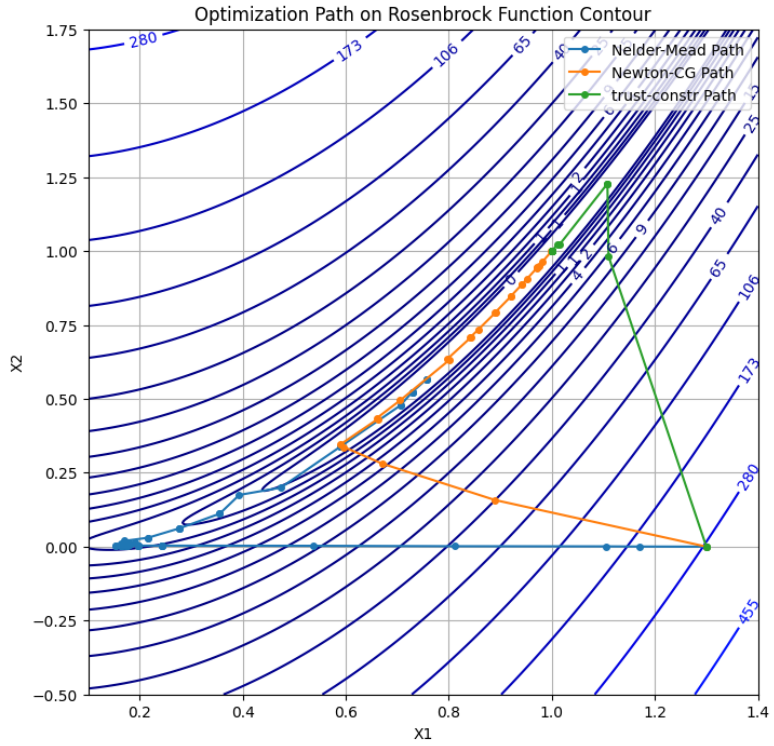

Figure 3: Contour map and the trace of three algorithms

We can see that the Trust-Region method converges faster than Nelder-Mead method and Newton-CG in this case with large condition number. And the convergence of Trust-Region shows superlinear convergence.

5.2 10D case

$$f(x) = \sum_{i=1}^{5} [100(x_{2i-1}^2 - x_{2i})^2 + (x_{2i-1} - 1)^2]$$

The function has one global minimizer $(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$.Take the start point randomly x_0 =(0.69646919, 0.28613933, 0.22685145, 0.55131477, 0.71946897, 0.42310646, 0.9807642 , 0.68482974, 0.4809319 , 0.39211752).

Compare the Trust-Region with gradient descent, Nelder-Mead method and Newton-CG.

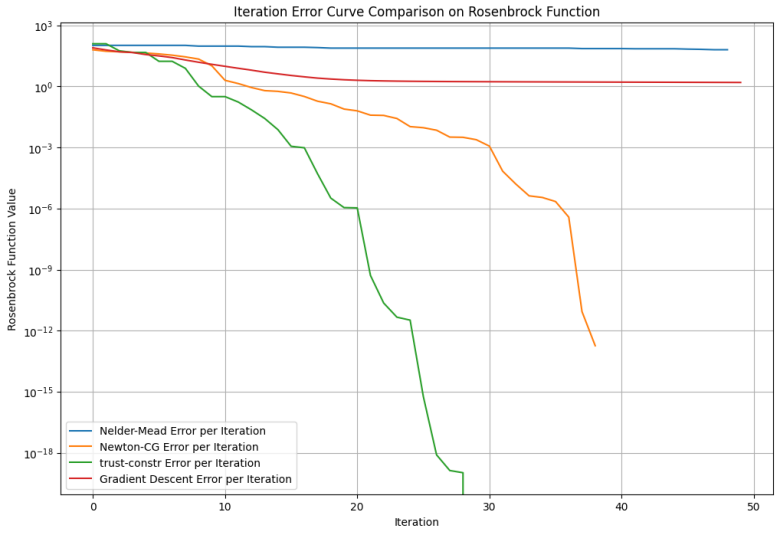

Figure 4: Iteration error of four algorithms(y-axis is log scale)

We can see that GD and N-M method converges to other local minimizer. They have large error eventually.Trust-Region and Newton-CG converge to the global minimizer with small error. But Trust-Region converges faster and more precise than Newton-CG. We can also see that the Trust-Region method can handle a large scale problem.

6 Conclusion and Acknowledgements

In this report, we introduce the idea of Trust-Region Method and its algorithm. We study the subproblems and find a way to solve it. By Cauchy point, we prove the global convergence and local superlinear convergence. Taking Rosenbrock function as an example of numerical Experiment, we can see the robust convergence of Trust-Region method and the faster speed of Trust-Region.

Over the course of my researching and writing this report, I would like to express my thanks to all those who have helped me.

I would like express my gratitude to all those who helped me during the writing of this thesis.

A special acknowledgement should be shown to Professor Li Lei, from whose lectures I benefited greatly. I learned a lot of optimization and it helps me solving practical problem.

I am particularly indebted to TAs gave me kind encouragement all through my writing and work hard to correct our homework and provide answers.

Sincere gratitude should also go to all my learned Professors and warm-hearted teachers who have greatly helped me in my study as well as in my life.

And my warm gratitude also goes to my friends and family who gave me much encouragement and financial support respectively. Moreover, I wish to extend my thanks to the library and the electronic reading room for their providing much useful information for my thesis.

7 Appendix

.1 2D Rosenbrock Function Experiment

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
from scipy.optimize import rosen, rosen_der, rosen_hess

#
x0 = np.array([1.3, 0])

#
methods = ['Nelder-Mead', 'Newton-CG', 'trust-constr']
errors = {method: [] for method in methods}
traces = {method: [] for method in methods}  #

def optimize_and_plot(method, jac=None, hess=None):
    def callback(xk, *args):
        errors[method].append(rosen(xk))
        traces[method].append(xk)  #

    res = minimize(rosen, x0, method=method,
                   jac=jac, hess=hess, callback=callback, options={'
                       maxiter': 50,'gtol': 1e-40})
    return res

#
res_nm = optimize_and_plot('Nelder-Mead')
res_ncg = optimize_and_plot('Newton-CG', jac=rosen_der, hess=
    rosen_hess)
res_trust_constr = optimize_and_plot('trust-constr', jac=rosen_der,
    hess=rosen_hess)

#
x = np.linspace(0.1, 1.4, 400)
y = np.linspace(-0.5, 1.75, 400)
X, Y = np.meshgrid(x, y)
Z = rosen([X, Y])

#
plt.figure(figsize=(8, 8))
contour = plt.contour(X, Y, Z, levels=np.logspace(-0.5, 3.5, 20),
    cmap='jet')
plt.clabel(contour, inline=1, fontsize=10)

#
for method, trace in traces.items():
    trace = np.array(trace)  #       numpy
```

```
    plt.plot(np.insert(trace[:, 0],0,x0[0]), np.insert(trace[:, 1],0,
        x0[1]), marker='o', markersize=4,label=f'{method} Path')
plt.title("Optimization Path on Rosenbrock Function Contour")
plt.xlabel("X1")
plt.ylabel("X2")
plt.legend()
plt.grid(True)
plt.show()
plt.figure(figsize=(6, 4))
for method in methods:
    plt.plot(errors[method], label=f'{method} Error per Iteration')
plt.title("Iteration Error Curve Comparison on 2D Rosenbrock Function
    ")
plt.yscale('log')
plt.xlabel("Iteration")
plt.ylabel("Rosenbrock Function Value")
plt.legend()
plt.grid(True)
plt.show()
```

.2 10D Rosenbrock Function Experiment

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize, line_search
from scipy.optimize import rosen, rosen_der, rosen_hess

#
dim = 10
np.random.seed(123)
x0 = np.random.rand(dim)  #       10

#
methods = ['Nelder-Mead', 'Newton-CG', 'trust-constr', 'Gradient
    Descent']
errors = {method: [] for method in methods}

def gradient_descent(func, grad_func, x0, callback=None, alpha0=1,
    max_iter=50, tol=1e-40):
    x = np.array(x0)
    for i in range(max_iter):
        gradient = grad_func(x)
        #
        search_dir = -gradient
        #
        line_result = line_search(func, grad_func, x, search_dir)
        alpha = line_result[0]
```

```
        if alpha is None:  #       line_search
            alpha = alpha0
        #
        x = x + alpha * search_dir
        #
        if callback is not None:
            callback(x)
        #
        if np.linalg.norm (search_dir) < tol:
            break
    return x

def optimize_and_plot(method, jac=None, hess=None):
    def callback(xk, *args):
        errors[method].append(rosen(xk))

    if method == 'Gradient Descent':
        gradient_descent(rosen, rosen_der, x0, callback=callback)
    else:
        minimize(rosen, x0, method=method,
                    jac=jac, hess=hess, callback=callback,
                    options={'maxiter': 50, 'gtol': 1e-40})

#     (Nelder-Mead)
optimize_and_plot('Nelder-Mead')

#    (Newton-CG)
optimize_and_plot('Newton-CG', jac=rosen_der, hess=rosen_hess)

#    (trust-constr)
optimize_and_plot('trust-constr', jac=rosen_der, hess=rosen_hess)

#     (Gradient Descent)
optimize_and_plot('Gradient Descent')

#
plt.figure(figsize=(12, 8))
for method in methods:
    plt.plot(errors[method], label=f'{method} Error per Iteration')
plt.title("Iteration Error Curve Comparison on Rosenbrock Function")
plt.yscale('log')
plt.xlabel("Iteration")
plt.ylabel("Rosenbrock Function Value")
plt.legend()
plt.grid(True)
plt.show()
```