

# 信任域方法

## 程序设计

基于信任域的牛顿法基本框架如参考教程算法4.1所示，核心步骤在于求解下式：

$$\min_{p \in \mathbb{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p \quad \text{s.t. } \|p\| \leq \Delta_k$$

得到在信任域半径内的最佳下降方向。在本次作业中，总共尝试了基于（1）柯西点；（2）Dogleg法；（3）子问题最优化（利用牛顿迭代求解 $\|p\| = \Delta$ ）来求解下降方向 $p_k$ 。

本次作业目标函数为 $n$ 元 Rosenbrock 函数，输入为 $n$ 元变量  $x$ ，返回值为该点处函数值，为方便求解，同时返回该点处的梯度值以及Hessian矩阵：

```
function [y, df, ddf] = Rosenbrock(x)
```

信任域总体框架函数定义如下：

```
function [x, y, count] = TrustRegion(f, f_p, x_0, delta_max, eta, max_iteration)
```

输入参数  $f$  为目标函数，本作业中为 $n$ 元 Rosenbrock 函数；参数  $f_p$  为不同的求解迭代方向的函数，本次实验中分别实现了 `CauchyPoint`，`Dogleg`，以及子问题最优化的 `Suboptimal` 三种，这三个函数形式如下：

```
function p = CauchyPoint(delta, g, b)
function p = Dogleg(delta, g, b)
function p = Suboptimal(delta, g, b)
```

输入参数均为当前信任域半径  $\delta$ ，该点处梯度  $g$ ，以及Hessian矩阵  $b$ ，返回值均为在不同准则下找到的最佳迭代方向  $p$ 。

参数  $x_0$  表示初始迭代点； $\delta_{\max}$  表示最大信任域半径； $\eta$  表示算法4.1中的 $\eta$ ； $\max\_iteration$  表示程序最大迭代次数。

返回值  $x$  表示信任域算法迭代的轨迹， $y$  记录迭代过程中的所有函数值， $count$  表示实际迭代次数。

信任域框架算法 `TrustRegion` 实现过程与书上算法4.1基本一致，此处略过。下面大致阐述三种迭代方向求解算法的实现思路。

1. 柯西点：求解在一阶梯度方向上的最优点：

$$p_k^c = \tau_k p_k^s$$
$$\text{其中: } p_k^s = -\frac{\Delta_k}{\|g_k\|} g_k$$
$$\tau_k = \begin{cases} 1 & \text{if } g_k^T B_k g_k \leq 0 \\ \min \left( \|g_k\|^3 / (\Delta_k g_k^T B_k g_k), 1 \right) & \text{otherwise.} \end{cases}$$

实现代码如下：

```

function p = CauchyPoint(delta, g, b)
temp = g' * b * g;
ps = -delta / norm(g) * g;
if temp <= 0
    tau = 1;
else
    tau = min((norm(g))^3 / (delta * temp), 1);
end
p = tau * ps;
end

```

2. Dogleg: 先沿一阶方向走到最优点, 在沿二阶方向走到最优点, 求该折线与信任域半径的交点:

1)  $p^*(\Delta) = p^B$  if  $\|p^B\| \leq \Delta$ , 否则

2) 求解:  $\|p(\tau)\| = \Delta$

$$p(\tau) = \begin{cases} \tau p^u, & 0 < \tau < 1 \\ p^u + (\tau - 1)(p^B - p^u), & 1 < \tau < 2 \end{cases}$$

$$\text{where } p^u = -\frac{g^T g}{g^T B g} g$$

实现代码如下:

```

function p = Dogleg(delta, g, b)
pb = -1 * b \ g;
pu = -1 * (g' * g) / (g' * b * g) * g;
if norm(pb) > delta
    fp = @(t) norm(t*pu) - delta; % 与第一条线段的交点
    t = double(solve(fp));
    if t > 0 && t < 1
        p = t*pu;
    else
        fp = @(t) norm((pu+(t-1)*(pb-pu))) - delta;
        t = double(solve(fp));
        t = max(t); % 这里应该有两个交点
        p = pu+(t-1)*(pb-pu);
    end
else
    p = pb;
end
end

```

这里实现的时候用了最简单粗暴的方式求折线与信任域半径的交点, 所以可能导致程序运行比较慢。

3. 子问题最优化:  $p(\lambda) = -(B + \lambda I)^{-1}g$ , 利用牛顿迭代法求解:

$$\phi(\lambda) = \|p(\lambda)\| - \Delta = 0$$

这里具体实现的时候参考书上算法4.3, 即利用Cholesky分解和牛顿迭代法来寻找最佳迭代方向  $p$ , 代码如下:

```

function p = Suboptimal(delta, g, b)
lam = 1e2; % 可以尽量大一下, 保证矩阵正定
max_iter = 5; % 一般迭代几次就差不多了
[m, n] = size(b);
for i = 1:max_iter
    [R, flag] = chol(b+lam*eye(m, n));
end

```

```

if flag > 0
    p = -b\g;
    break;
end
p = -(R'*R)\g;
q = R'\p;
lam = lam + (norm(p)/norm(q))^2*((norm(p)-delta)/delta);
if lam<10*eps
    p = -b\g;    % lam=0时，下降方向直接为牛顿方向，因为此时其不受信任域半径约束
    break;
end
end
end
end

```

## 运行结果

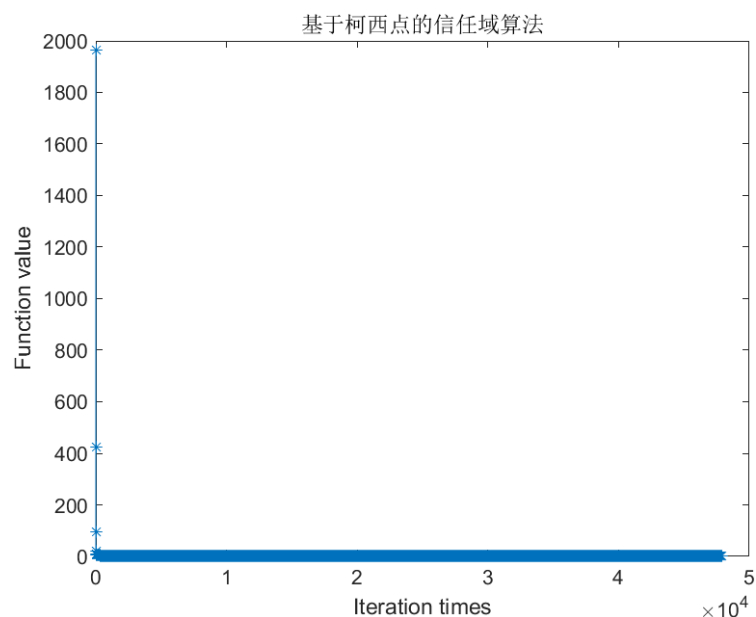
本次实验中以任意10元变量为初始点，为确保算法有效收敛，需使初始位置处的Hessian矩阵正定，代码如下：

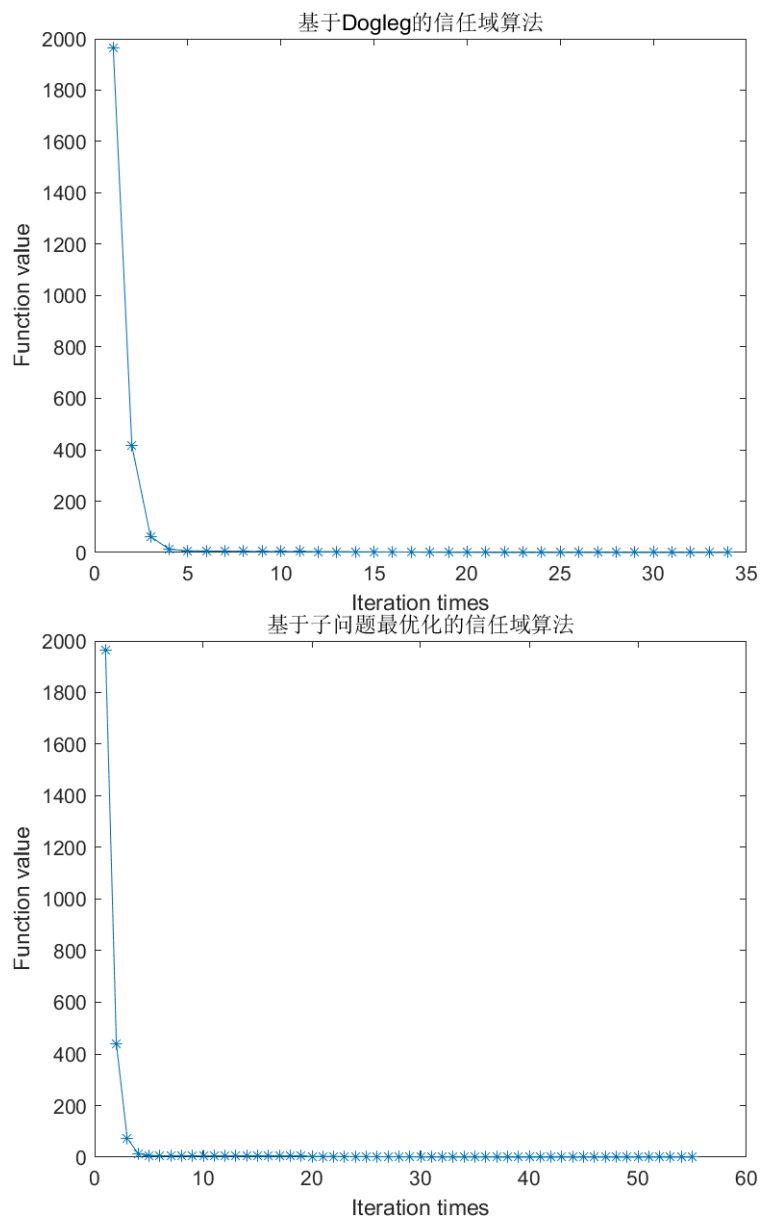
```

x0 = 3 * (rand(n, 1) - 0.5);    % 产生(-1.5, 1.5)的n维变量
[~, ~, B] = Rosenbrock(x0);
try
    [~, flag] = chol(B);        % 有可能初始值处Hessian矩阵不正当，不利于后续求解
catch ErrorInfo
    disp(ErrorInfo)
end
while flag > 0
    x0 = 3 * (rand(n, 1) - 0.5);
    [~, ~, B] = Rosenbrock(x0);
    [~, flag] = chol(B);
end

```

设置 `delta_max=2`, `eta=0.1`, `max_iteration=1e5`, 分别在三种不同 `f_p` 选项下调用 `TrustRegion` 函数，对迭代结果进行绘图，分别如下所示：



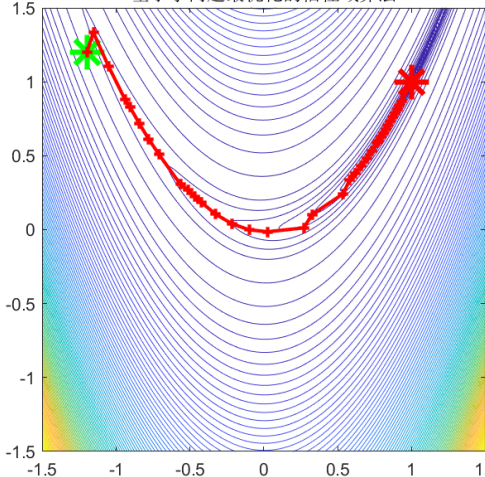
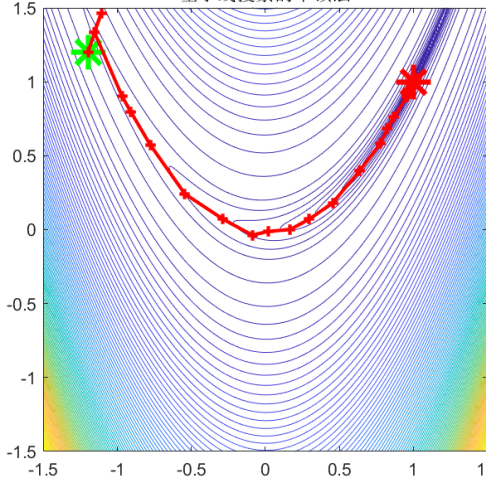


三种算法均收敛到了目标位置 $[1, 1, \dots, 1]^T$ 处，迭代次数分别如下表所示：

	CauchyPoint	Dogleg	SubOptimal
迭代次数	47907	33	54
最终函数值	1.9998e-26	0	0

由于柯西点算法实际只利用了一阶信息，属于线性收敛，所以其迭代次数最多，而Dogleg和Suboptimal都用到了二阶信息，所以收敛速度会更快一些。而程序运行时间的话，由于在函数实现的过程中采用了比较粗暴的方式，所以Dogleg花费时间最多，SubOptimal花费时间最少。经过我多次测试发现，上述不同算法的实际运行结果与初始值得选取有一定关系，所以上述比较不一定完全准确，但基本趋势应该一致。详细的实验结果请见main.mlx文件中。

下面是将信任域算法和线搜索算法的迭代轨迹进行可视化进行比较：

	信任域	线搜索
迭代轨迹	<p>基于子问题最优化的信任域算法</p> 	<p>基于线搜索的牛顿法</p> 
迭代次数	45	28
最终函数值	0	1.7810e-20

可以看到在该实例中虽然信任域算法迭代次数虽然比基于线搜索的牛顿法略微多一些，但是其最终却更好的收敛到了目标点。