# Practical Optimization Algorithms
# 实用优化算法

徐 翔

数学科学学院
浙江大学

Dec 2, 2021

# CHAPTER VI: TRUST-REGION METHODS (信赖域方法)

# Outline of the Trust-Region Approach

Line search methods and trust-region methods both generate steps with the help of a quadratic model（二次函数模型） of the objective function, but they use this model in different ways.

# Outline of the Trust-Region Approach

Line search methods and trust-region methods both generate steps with the help of a quadratic model（二次函数模型） of the objective function, but they use this model in different ways.

- Line search methods use it to generate a search direction （主要用于产生搜索方向）, and then focus their efforts on finding a suitable step length $\alpha$ along this direction.

# Outline of the Trust-Region Approach

Line search methods and trust-region methods both generate steps with the help of a quadratic model（二次函数模型） of the objective function, but they use this model in different ways.

- Line search methods use it to generate a search direction （主要用于产生搜索方向）, and then focus their efforts on finding a suitable step length $\alpha$ along this direction.
- Trust-region defines a region around the current iterate within which they trust the model to be an adequate representation of the objective function,

# OUTLINE OF THE TRUST-REGION APPROACH

Line search methods and trust-region methods both generate steps with the help of a quadratic model（二次函数模型） of the objective function, but they use this model in different ways.

- Line search methods use it to generate a search direction （主要用于产生搜索方向）, and then focus their efforts on finding a suitable step length $\alpha$ along this direction.
- Trust-region defines a region around the current iterate within which they trust the model to be an adequate representation of the objective function,
- then choose the step to be the approximate minimizer of the model in this region.

# Outline of the Trust-Region Approach

Line search methods and trust-region methods both generate steps with the help of a quadratic model（二次函数模型） of the objective function, but they use this model in different ways.

- Line search methods use it to generate a search direction （主要用于产生搜索方向）, and then focus their efforts on finding a suitable step length $\alpha$ along this direction.
- Trust-region defines a region around the current iterate within which they trust the model to be an adequate representation of the objective function,
- then choose the step to be the approximate minimizer of the model in this region.
- In effect, they choose the direction and length of the step simultaneously.

# Outline of the Trust-Region Approach

Line search methods and trust-region methods both generate steps with the help of a quadratic model（二次函数模型） of the objective function, but they use this model in different ways.

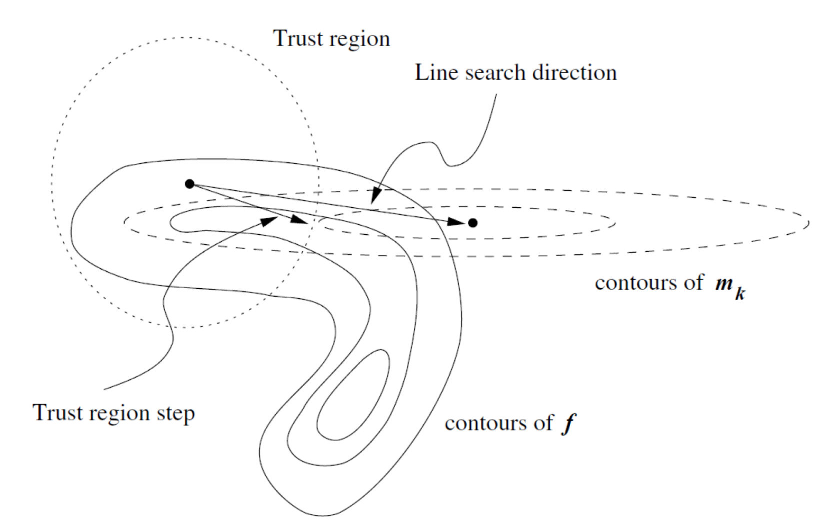
- Line search methods use it to generate a search direction （主要用于产生搜索方向）, and then focus their efforts on finding a suitable step length $\alpha$ along this direction.
- Trust-region defines a region around the current iterate within which they trust the model to be an adequate representation of the objective function,
- then choose the step to be the approximate minimizer of the model in this region.
- In effect, they choose the direction and length of the step simultaneously.
- If a step is not acceptable, they reduce the size of the region and find a new minimizer.

# OUTLINE OF THE TRUST-REGION APPROACH

Line search methods and trust-region methods both generate steps with the help of a quadratic model（二次函数模型） of the objective function, but they use this model in different ways.

- Line search methods use it to generate a search direction （主要用于产生搜索方向）, and then focus their efforts on finding a suitable step length $\alpha$ along this direction.
- Trust-region defines a region around the current iterate within which they trust the model to be an adequate representation of the objective function,
- then choose the step to be the approximate minimizer of the model in this region.
- In effect, they choose the direction and length of the step simultaneously.
- If a step is not acceptable, they reduce the size of the region and find a new minimizer.
- In general, the direction of the step changes whenever the size of the trust region is altered.

# Trust-region and line search steps



# Outline of the Trust-Region Approach

- In this chapter, we will assume that the model function $m_k$ that is used at each iteration $x_k$ is quadratic.

# Outline of the Trust-Region Approach

- In this chapter, we will assume that the model function $m_k$ that is used at each iteration $x_k$ is quadratic. Moreover $m_k$ is based on the Taylor-series expansion of $f$ around $x_k$ , which is

# Outline of the Trust-Region Approach

- In this chapter, we will assume that the model function $m_k$ that is used at each iteration $x_k$ is quadratic. Moreover $m_k$ is based on the Taylor-series expansion of $f$ around $x_k$ , which is

$$f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T \nabla f^2 (x_k + tp) p, \tag{6.1}$$

where $f_k = f(x_k), g_k = \nabla f(x_k)$ and $t \in (0, 1)$.

# Outline of the Trust-Region Approach

- In this chapter, we will assume that the model function $m_k$ that is used at each iteration $x_k$ is quadratic. Moreover $m_k$ is based on the Taylor-series expansion of $f$ around $x_k$ , which is

$$f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T \nabla f^2(x_k + tp)p, \tag{6.1}$$

  where $f_k = f(x_k), g_k = \nabla f(x_k)$ and $t \in (0, 1)$.

- By using an approximation $B_k$ to the Hessian in the second-order term,

# Outline of the Trust-Region Approach

- In this chapter, we will assume that the model function $m_k$ that is used at each iteration $x_k$ is quadratic. Moreover $m_k$ is based on the Taylor-series expansion of $f$ around $x_k$ , which is

$$f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T \nabla f^2 (x_k + tp) p, \tag{6.1}$$

  where $f_k = f(x_k), g_k = \nabla f(x_k)$ and $t \in (0, 1)$.

- By using an approximation $B_k$ to the Hessian in the second-order term, $m_k$ is defined as follows:

# Outline of the Trust-Region Approach

- In this chapter, we will assume that the model function $m_k$ that is used at each iteration $x_k$ is quadratic. Moreover $m_k$ is based on the Taylor-series expansion of $f$ around $x_k$ , which is

$$f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T \nabla f^2(x_k + tp)p, \tag{6.1}$$

  where $f_k = f(x_k), g_k = \nabla f(x_k)$ and $t \in (0, 1)$.

- By using an approximation $B_k$ to the Hessian in the second-order term, $m_k$ is defined as follows:

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p \tag{6.2}$$

# Outline of the Trust-Region Approach

- In this chapter, we will assume that the model function $m_k$ that is used at each iteration $x_k$ is quadratic. Moreover $m_k$ is based on the Taylor-series expansion of $f$ around $x_k$ , which is

$$f(x_k + p) = f_k + g_k^T p + \frac{1}{2} p^T \nabla f^2 (x_k + tp) p, \tag{6.1}$$

  where $f_k = f(x_k), g_k = \nabla f(x_k)$ and $t \in (0, 1)$.

- By using an approximation $B_k$ to the Hessian in the second-order term, $m_k$ is defined as follows:

$$m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p \tag{6.2}$$

  where $B_k$ is some symmetric matrix. The difference between $m_k(p)$ and $f(x_k + p)$ is $\mathcal{O}(\|p\|^2)$, which is small when $p$ is small.

# Outline of the Trust-Region Approach

- When $B_k = \nabla^2 f(x_k)$, the approximation error in the model function $m_k$ is $\mathcal{O}(\|p\|^2)$, so this model is especially accurate when $\|p\|$ is small.

# Outline of the Trust-Region Approach

- When $B_k = \nabla^2 f(x_k)$, the approximation error in the model function $m_k$ is $\mathcal{O}(\|p\|^2)$, so this model is especially accurate when $\|p\|$ is small. This choice $B_k = \nabla^2 f(x_k)$ leads to the trust-region Newton method.

# Outline of the Trust-Region Approach

- When $B_k = \nabla^2 f(x_k)$, the approximation error in the model function $m_k$ is $\mathcal{O}(\|p\|^2)$, so this model is especially accurate when $\|p\|$ is small. This choice $B_k = \nabla^2 f(x_k)$ leads to the trust-region Newton method.
- In the other part, we emphasis the generality of the trust-region approach by assuming little about $B_k$ except symmetry and uniform boundedness.

# Outline of the Trust-Region Approach

To obtain each step, we seek a solution of the subproblem

$$\min_{p \in \mathcal{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p, \quad s.t. \;\; \|p\| \leq \Delta_k \tag{6.3}$$

where $\Delta_k > 0$ is the trust-region radius.

# Outline of the Trust-Region Approach

The classification of trust-region methods are decided by the choice of $B_k$ and norm for trust region in the model (6.3). For example,

# Outline of the Trust-Region Approach

The classification of trust-region methods are decided by the choice of $B_k$ and norm for trust region in the model (6.3). For example,

- If $B_k = 0$ in (6.3) and define the trust region using the Euclidean norm, the trust-region method identifies with the steepest descent line search approach;

# Outline of the Trust-Region Approach

The classification of trust-region methods are decided by the choice of $B_k$ and norm for trust region in the model (6.3). For example,

- If $B_k = 0$ in (6.3) and define the trust region using the Euclidean norm, the trust-region method identifies with the steepest descent line search approach;
- If $B_k$ is chosen to be the exact Hessian $\nabla^2 f(x_k)$ , the resulting approach is called the trust-region Newton method;

# Outline of the Trust-Region Approach

The classification of trust-region methods are decided by the choice of $B_k$ and norm for trust region in the model (6.3). For example,

- If $B_k = 0$ in (6.3) and define the trust region using the Euclidean norm, the trust-region method identifies with the steepest descent line search approach;
- If $B_k$ is chosen to be the exact Hessian $\nabla^2 f(x_k)$ , the resulting approach is called the trust-region Newton method;
- If $B_k$ is defined by means of a quasi-Newton approximation, we obtain a trust-region quasi-Newton method.

# Outline of the Trust-Region Approach

The size of the trust region is critical to the effectiveness of each step.

- If the region is too small, the algorithm misses an opportunity to take a substantial step that will move it much closer to the minimizer of the objective function.
- If too large, the minimizer of the model may be far from the minimizer of the objective function in the region, so we may have to reduce the size of the region and try again.

# Outline of the Trust-Region Approach

In practical algorithms, we choose the size of the region according to the performance of the algorithm during previous iterations

# Outline of the Trust-Region Approach

In practical algorithms, we choose the size of the region according to the performance of the algorithm during previous iterations

- If the model is consistently reliable, producing good steps and accurately predicting the behavior of the objective function along these steps, the size of the trust region may be increased to allow longer, more ambitious, steps to be taken.

# Outline of the Trust-Region Approach

In practical algorithms, we choose the size of the region according to the performance of the algorithm during previous iterations

- If the model is consistently reliable, producing good steps and accurately predicting the behavior of the objective function along these steps, the size of the trust region may be increased to allow longer, more ambitious, steps to be taken.
- A failed step is an indication that our model is an inadequate representation of the objective function over the current trust region. After such a step, we reduce the size of the region and try again.

# Outline of the Trust-Region Approach

- Define the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \tag{6.4}$$

# Outline of the Trust-Region Approach

- Define the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \tag{6.4}$$

the numerator is called the actual reduction, and the denominator is the predicted reduction.

# Outline of the Trust-Region Approach

- Define the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \tag{6.4}$$

the numerator is called the actual reduction, and the denominator is the predicted reduction.

- Note that since the step $p_k$ is obtained by minimizing the model $m_k$ over a region that includes the step $p = 0$, the predicted reduction will always be nonnegative. Thus

# Outline of the Trust-Region Approach

- Define the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \tag{6.4}$$

  the numerator is called the actual reduction, and the denominator is the predicted reduction.

- Note that since the step $p_k$ is obtained by minimizing the model $m_k$ over a region that includes the step $p = 0$, the predicted reduction will always be nonnegative. Thus
  - if $\rho_k$ is negative, the new objective value $f(x_k + p_k)$ is greater than the current value $f(x_k)$, so the step must be rejected.

# Outline of the Trust-Region Approach

- Define the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \tag{6.4}$$

the numerator is called the actual reduction, and the denominator is the predicted reduction.

- Note that since the step $p_k$ is obtained by minimizing the model $m_k$ over a region that includes the step $p = 0$, the predicted reduction will always be nonnegative. Thus
  - if $\rho_k$ is negative, the new objective value $f(x_k + p_k)$ is greater than the current value $f(x_k)$, so the step must be rejected.
  - if $\rho_k$ is close to 1, there is good agreement between the model $m_k$ and the function $f$ over this step, so it is safe to expand the trust region for the next iteration.

# Outline of the Trust-Region Approach

- Define the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} \tag{6.4}$$

  the numerator is called the actual reduction, and the denominator is the predicted reduction.

- Note that since the step $p_k$ is obtained by minimizing the model $m_k$ over a region that includes the step $p = 0$, the predicted reduction will always be nonnegative. Thus
  - if $\rho_k$ is negative, the new objective value $f(x_k + p_k)$ is greater than the current value $f(x_k)$, so the step must be rejected.
  - if $\rho_k$ is close to 1, there is good agreement between the model $m_k$ and the function $f$ over this step, so it is safe to expand the trust region for the next iteration.
  - If $\rho_k$ is positive but not close to 1, we do not alter the trust region, but if it is close to zero or negative, we shrink the trust region.

# Outline of the Trust-Region Approach

Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$

# Outline of the Trust-Region Approach

Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$

# Outline of the Trust-Region Approach

Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
    Obtain $p_k$ by (approximately) solving (6.3);

# Outline of the Trust-Region Approach

### Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$

**for** $k = 0, 1, 2, \cdots$;

  Obtain $p_k$ by (approximately) solving (6.3);

  Evaluate $\rho_k$ from (6.4);

# Outline of the Trust-Region Approach

Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
Obtain $p_k$ by (approximately) solving (6.3);
Evaluate $\rho_k$ from (6.4);
**if** $\rho_k < \frac{1}{4}$;

# Outline of the Trust-Region Approach

## Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
    Obtain $p_k$ by (approximately) solving (6.3);
    Evaluate $\rho_k$ from (6.4);
    **if** $\rho_k < \frac{1}{4}$;
        $\Delta_{k+1} = \frac{1}{4}\Delta_k$

# Outline of the Trust-Region Approach

Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
Obtain $p_k$ by (approximately) solving (6.3);
Evaluate $\rho_k$ from (6.4);
**if** $\rho_k < \frac{1}{4}$;
$\Delta_{k+1} = \frac{1}{4}\Delta_k$
**else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$

# Outline of the Trust-Region Approach

Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
  Obtain $p_k$ by (approximately) solving (6.3);
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
    $\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$

# Outline of the Trust-Region Approach

Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
Obtain $p_k$ by (approximately) solving (6.3);
Evaluate $\rho_k$ from (6.4);
**if** $\rho_k < \frac{1}{4}$;
$\Delta_{k+1} = \frac{1}{4}\Delta_k$
**else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
$\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$
**else**

# Outline of the Trust-Region Approach

**Algorithm 1 (Trust Region)**

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
  Obtain $p_k$ by (approximately) solving (6.3);
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$
    **else**
      $\Delta_{k+1} = \Delta_k;$

# Outline of the Trust-Region Approach

Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
    Obtain $p_k$ by (approximately) solving (6.3);
    Evaluate $\rho_k$ from (6.4);
    **if** $\rho_k < \frac{1}{4}$;
        $\Delta_{k+1} = \frac{1}{4}\Delta_k$
    **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
            $\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$
        **else**
            $\Delta_{k+1} = \Delta_k;$
    **if** $\rho_k > \eta$

# Outline of the Trust-Region Approach

## Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
  Obtain $p_k$ by (approximately) solving (6.3);
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$
    **else**
      $\Delta_{k+1} = \Delta_k;$
  **if** $\rho_k > \eta$
    $x_{k+1} = x_k + p_k$

# Outline of the Trust-Region Approach

## Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$

Obtain $p_k$ by (approximately) solving (6.3);
Evaluate $\rho_k$ from (6.4);
**if** $\rho_k < \frac{1}{4}$;
$\Delta_{k+1} = \frac{1}{4}\Delta_k$
**else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
$\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$
**else**
$\Delta_{k+1} = \Delta_k;$
**if** $\rho_k > \eta$
$x_{k+1} = x_k + p_k$
**else**

# OUTLINE OF THE TRUST-REGION APPROACH

Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
  Obtain $p_k$ by (approximately) solving (6.3);
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$
    **else**
      $\Delta_{k+1} = \Delta_k$;
  **if** $\rho_k > \eta$
    $x_{k+1} = x_k + p_k$
  **else**
    $x_{k+1} = x_k$;

# Outline of the Trust-Region Approach

Algorithm 1 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in [0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
  Obtain $p_k$ by (approximately) solving (6.3);
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min(2\Delta_k, \hat{\Delta})$
    **else**
      $\Delta_{k+1} = \Delta_k;$
  **if** $\rho_k > \eta$
    $x_{k+1} = x_k + p_k$
  **else**
    $x_{k+1} = x_k;$
**end(for)**.

# Outline of the Trust-Region Approach

**Theorem**

*The vector $p^*$ is a global solution of the trust-region problem*

$$\min_{p\in\mathcal{R}^n} m(p) = f + g^T p + \frac{1}{2}p^T Bp, \quad s.t. \quad \|p\| \leq \Delta \tag{6.5}$$

# Outline of the Trust-Region Approach

**Theorem**

*The vector $p^*$ is a global solution of the trust-region problem*

$$\min_{p\in\mathcal{R}^n} m(p) = f + g^T p + \frac{1}{2}p^T B p, \quad s.t. \quad \|p\| \le \Delta \tag{6.5}$$

*if and only if*

# Outline of the Trust-Region Approach

**Theorem**

*The vector $p^*$ is a global solution of the trust-region problem*

$$\min_{p \in \mathcal{R}^n} m(p) = f + g^T p + \frac{1}{2} p^T B p, \quad s.t. \quad \|p\| \leq \Delta \tag{6.5}$$

*if and only if*

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

$$(B + \lambda I)p^* = -g, \tag{6.6a}$$

$$\lambda(\Delta - \|p^*\|) = 0, \tag{6.6b}$$

$$(B + \lambda I) \quad \text{is positive semi-definite.} \tag{6.6c}$$

# Outline of the Trust-Region Approach

- In most of our discussions, we define $\|\cdot\|$ to be the Euclidean norm, so that the solution $p_k^*$ of (6.3) is the minimizer of $m_k$ in the ball of radius $\Delta_k$.

# Outline of the Trust-Region Approach

- In most of our discussions, we define $\|\cdot\|$ to be the Euclidean norm, so that the solution $p_k^*$ of (6.3) is the minimizer of $m_k$ in the ball of radius $\Delta_k$.
- Thus, the trust-region approach requires us to solve a sequence of subproblems (6.3) in which the objective function and constraint (which can be written as $p^T p \leq \Delta_k^2$ ) are both quadratic.

# Outline of the Trust-Region Approach

- In most of our discussions, we define $\|\cdot\|$ to be the Euclidean norm, so that the solution $p_k^*$ of (6.3) is the minimizer of $m_k$ in the ball of radius $\Delta_k$.
- Thus, the trust-region approach requires us to solve a sequence of subproblems (6.3) in which the objective function and constraint (which can be written as $p^T p \leq \Delta_k^2$ ) are both quadratic.
- When $B_k$ is positive definite and $\|B_k^{-1} g_k\| \leq \Delta_k$, the solution of (6.3) is easy to identify - it is simply the unconstrained minimum $p_k^B = -B_k^{-1} g_k$ of the quadratic $m_k(p)$.

# Outline of the Trust-Region Approach

- In most of our discussions, we define $\|\cdot\|$ to be the Euclidean norm, so that the solution $p_k^*$ of (6.3) is the minimizer of $m_k$ in the ball of radius $\Delta_k$.
- Thus, the trust-region approach requires us to solve a sequence of subproblems (6.3) in which the objective function and constraint (which can be written as $p^T p \leq \Delta_k^2$ ) are both quadratic.
- When $B_k$ is positive definite and $\|B_k^{-1} g_k\| \leq \Delta_k$, the solution of (6.3) is easy to identify - it is simply the unconstrained minimum $p_k^B = -B_k^{-1} g_k$ of the quadratic $m_k(p)$.
- In this case, we call $p_k^B$ the *full step*.

# Outline of the Trust-Region Approach

- In most of our discussions, we define $\|\cdot\|$ to be the Euclidean norm, so that the solution $p_k^*$ of (6.3) is the minimizer of $m_k$ in the ball of radius $\Delta_k$.
- Thus, the trust-region approach requires us to solve a sequence of subproblems (6.3) in which the objective function and constraint (which can be written as $p^T p \leq \Delta_k^2$ ) are both quadratic.
- When $B_k$ is positive definite and $\|B_k^{-1} g_k\| \leq \Delta_k$, the solution of (6.3) is easy to identify - it is simply the unconstrained minimum $p_k^B = -B_k^{-1} g_k$ of the quadratic $m_k(p)$.
- In this case, we call $p_k^B$ the *full step*.
- The solution of (6.3) is not so obvious in other cases, but it usually be found without too much computational expense.

# Outline of the Trust-Region Approach

- In most of our discussions, we define $\|\cdot\|$ to be the Euclidean norm, so that the solution $p_k^*$ of (6.3) is the minimizer of $m_k$ in the ball of radius $\Delta_k$.
- Thus, the trust-region approach requires us to solve a sequence of subproblems (6.3) in which the objective function and constraint (which can be written as $p^T p \le \Delta_k^2$ ) are both quadratic.
- When $B_k$ is positive definite and $\|B_k^{-1} g_k\| \le \Delta_k$, the solution of (6.3) is easy to identify - it is simply the unconstrained minimum $p_k^B = -B_k^{-1} g_k$ of the quadratic $m_k(p)$.
- In this case, we call $p_k^B$ the *full step*.
- The solution of (6.3) is not so obvious in other cases, but it usually be found without too much computational expense.
- In any case, as described above, we need only an approximate solution to obtain convergence and good practical behavior.

# The Cauchy Point

# The Cauchy Point

- Intuition: line search methods do not require optimal step lengths to be globally convergent. In fact, only a crude approximation to the optimal step length that satisfies certain loose criteria is needed.

# The Cauchy Point

- Intuition: line search methods do not require optimal step lengths to be globally convergent. In fact, only a crude approximation to the optimal step length that satisfies certain loose criteria is needed.
- A similar situation applies in trust-region methods.

# The Cauchy Point

- Intuition: line search methods do not require optimal step lengths to be globally convergent. In fact, only a crude approximation to the optimal step length that satisfies certain loose criteria is needed.
- A similar situation applies in trust-region methods.
- Although in principle we are seeking the optimal solution of the subproblem (6.3), it is enough for global convergence purposes to find an approximate solution $p_k$ that lies within the trust region and gives a sufficient reduction in the model.

# THE CAUCHY POINT

- Intuition: line search methods do not require optimal step lengths to be globally convergent. In fact, only a crude approximation to the optimal step length that satisfies certain loose criteria is needed.
- A similar situation applies in trust-region methods.
- Although in principle we are seeking the optimal solution of the subproblem (6.3), it is enough for global convergence purposes to find an approximate solution $p_k$ that lies within the trust region and gives a sufficient reduction in the model.
- The sufficient reduction can be quantified in terms of the Cauchy point, which we denote by $p_k^C$ .

# The Cauchy Point

**Algorithm 2 (Cauchy Point Calculation)**

Find the vector $p_k^s$ that solves a linear version of (6.3), that is,

$$p_k^s = \arg\min_{p \in \mathcal{R}^n} f_k + g_k^T p, \quad s.t. \quad \|p\| \leq \Delta_k; \tag{6.7}$$

# The Cauchy Point

**Algorithm 2 (Cauchy Point Calculation)**

Find the vector $p_k^s$ that solves a linear version of (6.3), that is,

$$p_k^s = \arg\min_{p \in \mathcal{R}^n} f_k + g_k^T p, \quad s.t. \quad \|p\| \leq \Delta_k; \tag{6.7}$$

Calculate the scalar $\tau_k > 0$ that minimizes $m_k(\tau p_k^s)$ subject to satisfying the trust-region bound, that is

$$\tau_k = \arg\min_{\tau \geq 0} m_k(\tau p_k^s), \quad s.t. \quad \|\tau p_k^s\| \leq \Delta_k \tag{6.8}$$

# The Cauchy Point

**Algorithm 2 (Cauchy Point Calculation)**

Find the vector $p_k^s$ that solves a linear version of (6.3), that is,

$$p_k^s = \arg \min_{p \in \mathcal{R}^n} f_k + g_k^T p, \quad s.t. \quad \|p\| \leq \Delta_k; \tag{6.7}$$

Calculate the scalar $\tau_k > 0$ that minimizes $m_k(\tau p_k^s)$ subject to satisfying the trust-region bound, that is

$$\tau_k = \arg \min_{\tau \geq 0} m_k(\tau p_k^s), \quad s.t. \quad \|\tau p_k^s\| \leq \Delta_k \tag{6.8}$$

Set

$$p_k^C = \tau_k p_k^s$$

.

# The Cauchy Point

# The Cauchy Point

- The solution of (6.7) is simply

$$p_k^s = -\frac{\Delta_k}{\|\nabla f(x_k)\|}\nabla f(x_k)$$

# THE CAUCHY POINT

- The solution of (6.7) is simply

$$p_k^s = -\frac{\Delta_k}{\|\nabla f(x_k)\|}\nabla f(x_k)$$

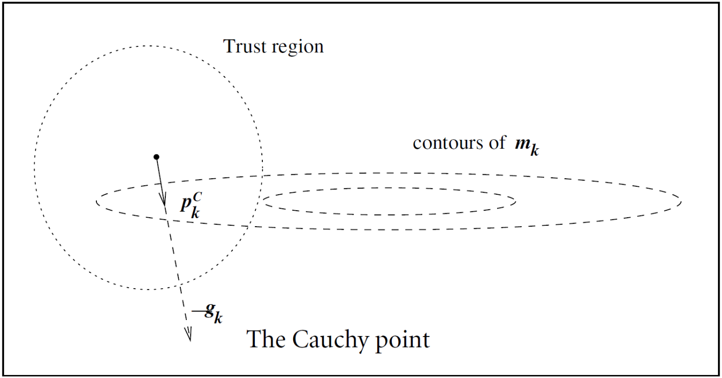
- Furthermore, a closed-form definition of the Cauchy point can be written in the following

$$p_k^C = -\tau_k \frac{\Delta_k}{\|\nabla f(x_k)\|}\nabla f(x_k), \tag{6.9}$$

where

$$\tau_k = \begin{cases} 1, & \text{if } \nabla f(x_k)^T B_k \nabla f(x_k) \le 0 \\ \min\left\{\|\nabla f(x_k)\|^3/(\nabla f(x_k)^T B_k \nabla f(x_k)), 1\right\} & \text{otherwise.} \end{cases} \tag{6.10}$$

# Cauchy Point For A Subproblem



# The Cauchy Point

# The Cauchy Point

- The Cauchy step $p_k^C$ is inexpensive to calculate -

# The Cauchy Point

- The Cauchy step $p_k^C$ is inexpensive to calculate - no matrix factorizations are required -

# The Cauchy Point

- The Cauchy step $p_k^C$ is inexpensive to calculate - no matrix factorizations are required - and is of crucial importance in deciding if an approximate solution of the trust-region subproblem is acceptable.

# The Cauchy Point

- The Cauchy step $p_k^C$ is inexpensive to calculate - no matrix factorizations are required - and is of crucial importance in deciding if an approximate solution of the trust-region subproblem is acceptable.
- Specifically, a trust-region method will be globally convergent if its steps $p_k$ attain a sufficient reduction in $m_k$; that is,

# The Cauchy Point

- The Cauchy step $p_k^C$ is inexpensive to calculate - no matrix factorizations are required - and is of crucial importance in deciding if an approximate solution of the trust-region subproblem is acceptable.
- Specifically, a trust-region method will be globally convergent if its steps $p_k$ attain a sufficient reduction in $m_k$; that is, they give a reduction in the model $m_k$ that is at least some fixed multiple of the decrease attained by the Cauchy step at each iteration.

# The Cauchy Point

- The Cauchy step $p_k^C$ is inexpensive to calculate - no matrix factorizations are required - and is of crucial importance in deciding if an approximate solution of the trust-region subproblem is acceptable.
- Specifically, a trust-region method will be globally convergent if its steps $p_k$ attain a sufficient reduction in $m_k$; that is, they give a reduction in the model $m_k$ that is at least some fixed multiple of the decrease attained by the Cauchy step at each iteration.
- The model reduction obtained by the Cauchy point is

$$m_k(0) - m_k(p_k^C) \leq \frac{1}{2}\|\nabla f(x_k)\| \min\left\{\Delta_k, \frac{\|\nabla f(x_k)\|}{\|B_k\|}\right\}$$

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots;$

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
Calculate $p_k = p_k^C$;

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
Calculate $p_k = p_k^C$;
Evaluate $\rho_k$ from (6.4);

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$

# The Cauchy Point

## Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots;$
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$

$$\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$$

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
    $$\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$$
    **else**

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
    $\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$
  **else**
    $\Delta_{k+1} = \Delta_k$;

# The Cauchy Point

## Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
    Calculate $p_k = p_k^C$;
    Evaluate $\rho_k$ from (6.4);
    **if** $\rho_k < \frac{1}{4}$;
        $\Delta_{k+1} = \frac{1}{4}\Delta_k$
    **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
            $\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$
        **else**
            $\Delta_{k+1} = \Delta_k$;
    **if** $\rho_k > \eta = 0$

# The Cauchy Point

## Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$
    **else**
      $\Delta_{k+1} = \Delta_k$;
  **if** $\rho_k > \eta = 0$
    $x_{k+1} = x_k + p_k$

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$
    **else**
      $\Delta_{k+1} = \Delta_k$;
  **if** $\rho_k > \eta = 0$
    $x_{k+1} = x_k + p_k$
  **else**

# The Cauchy Point

## Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots;$
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$
    **else**
      $\Delta_{k+1} = \Delta_k;$
  **if** $\rho_k > \eta = 0$
    $x_{k+1} = x_k + p_k$
  **else**
    $x_{k+1} = x_k;$

# The Cauchy Point

Algorithm 3 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta = 0$
**for** $k = 0, 1, 2, \cdots$;
Calculate $p_k = p_k^C$;
Evaluate $\rho_k$ from (6.4);
**if** $\rho_k < \frac{1}{4}$;
$\Delta_{k+1} = \frac{1}{4}\Delta_k$
**else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
$\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$
**else**
$\Delta_{k+1} = \Delta_k$;
**if** $\rho_k > \eta = 0$
$x_{k+1} = x_k + p_k$
**else**
$x_{k+1} = x_k$;
**end(for)**.

# Convergence to Stationary Points

**Theorem**

- Suppose that $B_k \leq \beta$ for some constant $\beta$,

# Convergence to Stationary Points

**Theorem**

- Suppose that $B_k \leq \beta$ for some constant $\beta$,
- $f$ is continuously differentiable and bounded below on the level set

$$\left\{x \middle| f(x) \leq f(x_0)\right\},$$

# Convergence to Stationary Points

**Theorem**

- Suppose that $B_k \leq \beta$ for some constant $\beta$,
- $f$ is continuously differentiable and bounded below on the level set

$$\left\{x \middle| f(x) \leq f(x_0)\right\},$$

- Then,

# Convergence to Stationary Points

### Theorem

- Suppose that $B_k \leq \beta$ for some constant $\beta$,
- $f$ is continuously differentiable and bounded below on the level set

$$\left\{x \middle| f(x) \leq f(x_0)\right\},$$

- Then, if above algorithm is not terminate in finite steps, we have

$$\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0.$$

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
Calculate $p_k = p_k^C$;

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
    Calculate $p_k = p_k^C$;
    Evaluate $\rho_k$ from (6.4);

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
Calculate $p_k = p_k^C$;
Evaluate $\rho_k$ from (6.4);
**if** $\rho_k < \frac{1}{4}$;

# The Cauchy Point

### Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
Calculate $p_k = p_k^C$;
Evaluate $\rho_k$ from (6.4);
**if** $\rho_k < \frac{1}{4}$;
$\Delta_{k+1} = \frac{1}{4}\Delta_k$
**else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
$$\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$$

# The Cauchy Point

## Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
    $$\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$$
    **else**

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots;$
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
    $\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$
  **else**
    $\Delta_{k+1} = \Delta_k;$

# The Cauchy Point

### Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$
    **else**
      $\Delta_{k+1} = \Delta_k$;
  **if** $\rho_k > \eta$

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$
    **else**
      $\Delta_{k+1} = \Delta_k$;
  **if** $\rho_k > \eta$
    $x_{k+1} = x_k + p_k$

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$

**for** $k = 0, 1, 2, \cdots;$

Calculate $p_k = p_k^C$;

Evaluate $\rho_k$ from (6.4);

**if** $\rho_k < \frac{1}{4}$;

$\Delta_{k+1} = \frac{1}{4}\Delta_k$

**else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$

$\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$

**else**

$\Delta_{k+1} = \Delta_k;$

**if** $\rho_k > \eta$

$x_{k+1} = x_k + p_k$

**else**

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$
**for** $k = 0, 1, 2, \cdots$;
  Calculate $p_k = p_k^C$;
  Evaluate $\rho_k$ from (6.4);
  **if** $\rho_k < \frac{1}{4}$;
    $\Delta_{k+1} = \frac{1}{4}\Delta_k$
  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$
      $\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$
    **else**
      $\Delta_{k+1} = \Delta_k$;
  **if** $\rho_k > \eta$
    $x_{k+1} = x_k + p_k$
  **else**
    $x_{k+1} = x_k$;

# The Cauchy Point

Algorithm 4 (Trust Region)

**Given** $\hat{\Delta} > 0$, $\Delta_0 \in (0, \hat{\Delta})$, and $\eta \in (0, \frac{1}{4})$

**for** $k = 0, 1, 2, \cdots$;

  Calculate $p_k = p_k^C$;

  Evaluate $\rho_k$ from (6.4);

  **if** $\rho_k < \frac{1}{4}$;

    $\Delta_{k+1} = \frac{1}{4}\Delta_k$

  **else if** $\rho_k > \frac{3}{4}$ and $\|p_k\| = \Delta_k$

      $\Delta_{k+1} = \min\left\{2\Delta_k, \hat{\Delta}\right\}$

    **else**

      $\Delta_{k+1} = \Delta_k$;

  **if** $\rho_k > \eta$

    $x_{k+1} = x_k + p_k$

  **else**

    $x_{k+1} = x_k$;

**end(for)**.

# Convergence to Stationary Points

**Theorem**

- Suppose that $B_k \leq \beta$ for some constant $\beta$,

# Convergence to Stationary Points

**Theorem**

- Suppose that $B_k \leq \beta$ for some constant $\beta$,
- $f$ is continuously differentiable and bounded below on the level set

$$\left\{ x \middle| f(x) \leq f(x_0) \right\},$$

# Convergence to Stationary Points

**Theorem**

- Suppose that $B_k \leq \beta$ for some constant $\beta$,
- $f$ is continuously differentiable and bounded below on the level set

$$\left\{ x \middle| f(x) \leq f(x_0) \right\},$$

- Then,

# Convergence to Stationary Points

**Theorem**

- Suppose that $B_k \leq \beta$ for some constant $\beta$,
- $f$ is continuously differentiable and bounded below on the level set

$$\left\{ x \middle| f(x) \leq f(x_0) \right\},$$

- Then, if above algorithm is not terminate in finite steps, we have

$$\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0.$$

# Improve the Cauchy Point

- Since the Cauchy point $p_k^C$ provides sufficient reduction in the model function $m_k$ to yield global convergence,

# Improve the Cauchy Point

- Since the Cauchy point $p_k^C$ provides sufficient reduction in the model function $m_k$ to yield global convergence, and since the cost of calculating it is so small,

# Improve the Cauchy Point

- Since the Cauchy point $p_k^C$ provides sufficient reduction in the model function $m_k$ to yield global convergence, and since the cost of calculating it is so small, why should we look any further for a better approximate solution of (6.3)?

# Improve the Cauchy Point

- Since the Cauchy point $p_k^C$ provides sufficient reduction in the model function $m_k$ to yield global convergence, and since the cost of calculating it is so small, why should we look any further for a better approximate solution of (6.3)?
- The reason is that by always taking the Cauchy point as our step, we are simply implementing the steepest descent method with a particular choice of step length.

# IMPROVE THE CAUCHY POINT

- Since the Cauchy point $p_k^C$ provides sufficient reduction in the model function $m_k$ to yield global convergence, and since the cost of calculating it is so small, why should we look any further for a better approximate solution of (6.3)?
- The reason is that by always taking the Cauchy point as our step, we are simply implementing the steepest descent method with a particular choice of step length.
- Since steepest descent performs poorly even if an optimal step length is used at each iteration, to make the Trust Region algorithm efficient in practice, we have to improve on the Cauchy point.

# Improve the Cauchy Point

# Improve the Cauchy Point

- Since we will be focusing on the internal workings of a single iteration, so we drop the subscript $k$ from the quantities to simplify the notation

# Improve the Cauchy Point

- Since we will be focusing on the internal workings of a single iteration, so we drop the subscript $k$ from the quantities to simplify the notation
- With this simplification, we restate the trust-region subproblem as follows:

$$\min_{p\in\mathcal{R}^n} m(p) = f + g^T p + \frac{1}{2} p^T B p, \quad s.t. \quad \|p\| \leq \Delta \tag{6.11}$$

# Improve the Cauchy Point

- Since we will be focusing on the internal workings of a single iteration, so we drop the subscript $k$ from the quantities to simplify the notation
- With this simplification, we restate the trust-region subproblem as follows:

$$\min_{p \in \mathcal{R}^n} m(p) = f + g^T p + \frac{1}{2} p^T B p, \quad s.t. \quad \|p\| \leq \Delta \tag{6.11}$$

- We denote the solution of above problem by $p^*(\Delta)$, to emphasize the dependence on $\Delta$.

# Improve the Cauchy Point

# Improve the Cauchy Point

- A number of algorithms for generating approximate solutions $p_k$ to the trust-region problem (6.11) start by computing the Cauchy point and then try to improve on it.

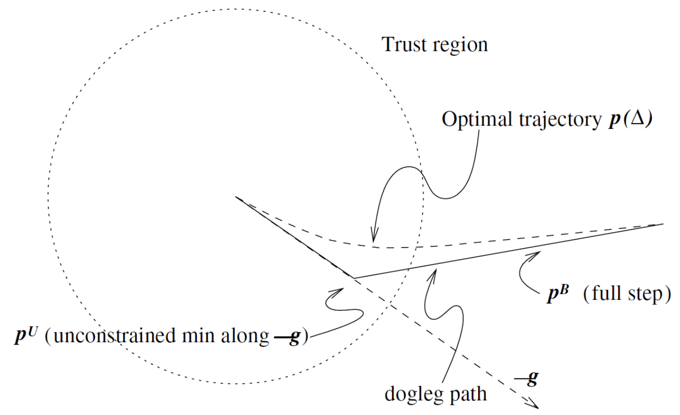
# Improve the Cauchy Point

- A number of algorithms for generating approximate solutions $p_k$ to the trust-region problem (6.11) start by computing the Cauchy point and then try to improve on it.
- The improvement strategy is often designed so that the full step $p_B = -B^{-1}g$ is chosen whenever $B$ is positive definite and $\|p_B\| \leq \Delta$.

# Improve the Cauchy Point

- A number of algorithms for generating approximate solutions $p_k$ to the trust-region problem (6.11) start by computing the Cauchy point and then try to improve on it.
- The improvement strategy is often designed so that the full step $p_B = -B^{-1}g$ is chosen whenever $B$ is positive definite and $\|p_B\| \leq \Delta$.
- When $B_k$ is the exact Hessian or a quasi-Newton approximation, this strategy can be expected to yield superlinear convergence.

# The Dogleg Method




Exact trajectory and dogleg approximation

# The Dogleg Method

$$\tau_k = \begin{cases} \tau p^U, & 0 \leq \tau \leq 1, \\ p^U + (\tau - 1)(p^B - p^U), & 1 \leq \tau \leq 2 \end{cases} \qquad (6.12)$$

where

$$p^U = -\frac{g^T g}{g^T B g} g$$

is the unconstrained minimizer of $m(\cdot)$ along the steepest descent direction.

# The Dogleg Method

- The dogleg method chooses $p$ to minimize the model $m$ along this path, subject to the trust-region bound.

# The Dogleg Method

- The dogleg method chooses $p$ to minimize the model $m$ along this path, subject to the trust-region bound.
- In fact, it is not even necessary to carry out a search, because the dogleg path intersects the trust-region boundary at most once and the intersection point can be computed analytically.The following theorem proves these claims.

# The Dogleg Method

- The dogleg method chooses $p$ to minimize the model $m$ along this path, subject to the trust-region bound.
- In fact, it is not even necessary to carry out a search, because the dogleg path intersects the trust-region boundary at most once and the intersection point can be computed analytically. The following theorem proves these claims.

**Theorem**

# The Dogleg Method

- The dogleg method chooses $p$ to minimize the model $m$ along this path, subject to the trust-region bound.
- In fact, it is not even necessary to carry out a search, because the dogleg path intersects the trust-region boundary at most once and the intersection point can be computed analytically.The following theorem proves these claims.

**Theorem**

Let $B$ be positive definite. Then

# The Dogleg Method

- The dogleg method chooses $p$ to minimize the model $m$ along this path, subject to the trust-region bound.
- In fact, it is not even necessary to carry out a search, because the dogleg path intersects the trust-region boundary at most once and the intersection point can be computed analytically.The following theorem proves these claims.

**Theorem**

Let $B$ be positive definite. Then

- $\|\tilde{p}(\tau)\|$ is an increasing function of $\tau$, and

# The Dogleg Method

- The dogleg method chooses $p$ to minimize the model $m$ along this path, subject to the trust-region bound.
- In fact, it is not even necessary to carry out a search, because the dogleg path intersects the trust-region boundary at most once and the intersection point can be computed analytically.The following theorem proves these claims.

**Theorem**

Let $B$ be positive definite. Then

- $\|\tilde{p}(\tau)\|$ is an increasing function of $\tau$, and
- $m(\tilde{p}(\tau))$ is a decreasing function of $\tau$.

# The Dogleg Method

# The Dogleg Method

- It follows from above theorem that the path $\tilde{p}(\tau)$ intersects the trust-region boundary $\|p\| = \Delta$ at exactly one point if $\|p^B\| \geq \Delta$, and nowhere otherwise.

# The Dogleg Method

- It follows from above theorem that the path $\tilde{p}(\tau)$ intersects the trust-region boundary $\|p\| = \Delta$ at exactly one point if $\|p^B\| \geq \Delta$, and nowhere otherwise.
- Since $m$ is decreasing along the path, the chosen value of $p$ will be at $p^B$ if $\|p^B\| \leq \Delta$,

# The Dogleg Method

- It follows from above theorem that the path $\tilde{p}(\tau)$ intersects the trust-region boundary $\|p\| = \Delta$ at exactly one point if $\|p^B\| \geq \Delta$, and nowhere otherwise.
- Since $m$ is decreasing along the path, the chosen value of $p$ will be at $p^B$ if $\|p^B\| \leq \Delta$, otherwise at the point of intersection of the dogleg and the trust-region boundary.

# The Dogleg Method

- It follows from above theorem that the path $\tilde{p}(\tau)$ intersects the trust-region boundary $\|p\| = \Delta$ at exactly one point if $\|p^B\| \geq \Delta$, and nowhere otherwise.
- Since $m$ is decreasing along the path, the chosen value of $p$ will be at $p^B$ if $\|p^B\| \leq \Delta$, otherwise at the point of intersection of the dogleg and the trust-region boundary.
- In the latter case, we compute the appropriate value of $\tau$ by solving the following scalar quadratic equation:

$$\|p^U + (\tau - 1)(p^B - p^U)\|^2 = \Delta^2.$$

# Two-Dimensional Subspace Minimization

- When $B$ is positive definite,

# Two-Dimensional Subspace Minimization

- When $B$ is positive definite, the dogleg method strategy can be made slightly more sophisticated by widening the search for $p$ to the entire two-dimensional subspace spanned by $p^U$ and $p^B$(equivalently, $g$ and $-B^{-1}g$).

# Two-Dimensional Subspace Minimization

- When $B$ is positive definite, the dogleg method strategy can be made slightly more sophisticated by widening the search for $p$ to the entire two-dimensional subspace spanned by $p^U$ and $p^B$(equivalently, $g$ and $-B^{-1}g$).
- The subproblem could be replaced by

# Two-Dimensional Subspace Minimization

- When $B$ is positive definite, the dogleg method strategy can be made slightly more sophisticated by widening the search for $p$ to the entire two-dimensional subspace spanned by $p^U$ and $p^B$(equivalently, $g$ and $-B^{-1}g$).
- The subproblem could be replaced by

$$\min_p m(p) = f + g^T p + \frac{1}{2} p^T B p, \quad s.t. \quad \|p\| \leq \Delta, p \in [g, B^{-1}g].$$

# Two-Dimensional Subspace Minimization

- When $B$ is positive definite, the dogleg method strategy can be made slightly more sophisticated by widening the search for $p$ to the entire two-dimensional subspace spanned by $p^U$ and $p^B$(equivalently, $g$ and $-B^{-1}g$).
- The subproblem could be replaced by

$$\min_{p} m(p) = f + g^T p + \frac{1}{2} p^T B p, \quad s.t. \quad \|p\| \le \Delta, p \in [g, B^{-1}g].$$

- This is a problem in two variables that is computationally inexpensive to solve.

# Two-Dimensional Subspace Minimization

- When $B$ is positive definite, the dogleg method strategy can be made slightly more sophisticated by widening the search for $p$ to the entire two-dimensional subspace spanned by $p^U$ and $p^B$(equivalently, $g$ and $-B^{-1}g$).
- The subproblem could be replaced by

$$\min_p m(p) = f + g^T p + \frac{1}{2} p^T B p, \quad s.t. \quad \|p\| \leq \Delta, p \in [g, B^{-1}g].$$

- This is a problem in two variables that is computationally inexpensive to solve.
- After some algebraic manipulation it can be reduced to finding the roots of a fourth degree polynomial.

# Steihaug's Approach

Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;

# Steihaug's Approach

Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;

# Steihaug's Approach

Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
  **return** $p = p_0$;

# Steihaug's Approach

Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
    **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$

# Steihaug's Approach

Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
    **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$
    **if** $d_j^T B d_j \leq 0$;

# Steihaug's Approach

Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
    **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$
    **if** $d_j^T B d_j \leq 0$;
        Find $\tau$ such that $p = p_j + \tau d_j$ minimizes $m(p)$ and satisfies $\|p\| = \Delta$;
        **return** $p$;

# Steihaug's Approach

## Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
    **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$
    **if** $d_j^T B d_j \leq 0$;
        Find $\tau$ such that $p = p_j + \tau d_j$ minimizes $m(p)$ and satisfies $\|p\| = \Delta$;
        **return** $p$;
    **else**

# Steihaug's Approach

### Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
    **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$
    **if** $d_j^T B d_j \leq 0$;
        Find $\tau$ such that $p = p_j + \tau d_j$ minimizes $m(p)$ and satisfies $\|p\| = \Delta$;
        **return** $p$;
    **else**
        Set $\alpha_j = r_j^T r_j / (d_j^T B d_j)$; Set $p_{j+1} = p_j + \alpha_j d_j$;

# Steihaug's Approach

## Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
  **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$
  **if** $d_j^T B d_j \leq 0$;
    Find $\tau$ such that $p = p_j + \tau d_j$ minimizes $m(p)$ and satisfies $\|p\| = \Delta$;
    **return** $p$;
  **else**
    Set $\alpha_j = r_j^T r_j / (d_j^T B d_j)$; Set $p_{j+1} = p_j + \alpha_j d_j$;
    **if** $\|p_{j+1}\| \geq \Delta$

# Steihaug's Approach

Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
  **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$
  **if** $d_j^T B d_j \leq 0$;
    Find $\tau$ such that $p = p_j + \tau d_j$ minimizes $m(p)$ and satisfies $\|p\| = \Delta$;
    **return** $p$;
  **else**
    Set $\alpha_j = r_j^T r_j / (d_j^T B d_j)$; Set $p_{j+1} = p_j + \alpha_j d_j$;
    **if** $\|p_{j+1}\| \geq \Delta$
      Find $\tau \geq 0$ such that $p = p_j + \tau d_j$ satisfies $\|p\| = \Delta$ **return** $p$;

# Steihaug's Approach

Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \le \epsilon$;
  **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$
  **if** $d_j^T B d_j \le 0$;
    Find $\tau$ such that $p = p_j + \tau d_j$ minimizes $m(p)$ and satisfies $\|p\| = \Delta$;
    **return** $p$;
  **else**
    Set $\alpha_j = r_j^T r_j / (d_j^T B d_j)$; Set $p_{j+1} = p_j + \alpha_j d_j$;
    **if** $\|p_{j+1}\| \ge \Delta$
      Find $\tau \ge 0$ such that $p = p_j + \tau d_j$ satisfies $\|p\| = \Delta$ **return** $p$;
      **else** Set $r_{j+1} = r_j + \alpha_j B d_j$;

# STEIHAUG'S APPROACH

## Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
  **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$
  **if** $d_j^T B d_j \leq 0$;
    Find $\tau$ such that $p = p_j + \tau d_j$ minimizes $m(p)$ and satisfies $\|p\| = \Delta$;
    **return** $p$;
  **else**
    Set $\alpha_j = r_j^T r_j / (d_j^T B d_j)$; Set $p_{j+1} = p_j + \alpha_j d_j$;
    **if** $\|p_{j+1}\| \geq \Delta$
      Find $\tau \geq 0$ such that $p = p_j + \tau d_j$ satisfies $\|p\| = \Delta$ **return** $p$;
      **else** Set $r_{j+1} = r_j + \alpha_j B d_j$;
      **if** $\|r_{j+1}\| \leq \epsilon \|r_0\|$ **return** $p = p_{j+1}$

# Steihaug's Approach

## Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
  **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$
  **if** $d_j^T B d_j \leq 0$;
    Find $\tau$ such that $p = p_j + \tau d_j$ minimizes $m(p)$ and satisfies $\|p\| = \Delta$;
    **return** $p$;
  **else**
    Set $\alpha_j = r_j^T r_j / (d_j^T B d_j)$; Set $p_{j+1} = p_j + \alpha_j d_j$;
    **if** $\|p_{j+1}\| \geq \Delta$
      Find $\tau \geq 0$ such that $p = p_j + \tau d_j$ satisfies $\|p\| = \Delta$ **return** $p$;
      **else** Set $r_{j+1} = r_j + \alpha_j B d_j$;
      **if** $\|r_{j+1}\| \leq \epsilon \|r_0\|$ **return** $p = p_{j+1}$
      **else**
      Set $\beta_{j+1} = r_{j+1}^T r_{j+1} / (r_j^T r_j)$, $d_{j+1} = r_{j+1} + \beta_{j+1} d_j$;

# Steihaug's Approach

## Algorithm 5 (CG-Steihaug)

**Given** $\epsilon > 0$; **Set** $p_0 = 0$, $r_0 = g$, $d_0 = -r_0$;
**if** $\|r_0\| \leq \epsilon$;
    **return** $p = p_0$;
**for** $j = 0, 1, 2, \cdots$
    **if** $d_j^T B d_j \leq 0$;
        Find $\tau$ such that $p = p_j + \tau d_j$ minimizes $m(p)$ and satisfies $\|p\| = \Delta$;
        **return** $p$;
    **else**
        Set $\alpha_j = r_j^T r_j / (d_j^T B d_j)$; Set $p_{j+1} = p_j + \alpha_j d_j$;
        **if** $\|p_{j+1}\| \geq \Delta$
            Find $\tau \geq 0$ such that $p = p_j + \tau d_j$ satisfies $\|p\| = \Delta$ **return** $p$;
            **else** Set $r_{j+1} = r_j + \alpha_j B d_j$;
            **if** $\|r_{j+1}\| \leq \epsilon \|r_0\|$ **return** $p = p_{j+1}$
            **else**
            Set $\beta_{j+1} = r_{j+1}^T r_{j+1} / (r_j^T r_j)$, $d_{j+1} = r_{j+1} + \beta_{j+1} d_j$;
**end(for)**.

# Reduction Obtained by The Cauchy Point

**Lemma**

The dogleg and two dimensional subspace minimization algorithms produce approximate solution $p_k$ of the subproblem (3) that satisfy the following estimate of decrease in the model function:

$$m_k(0) - m_k(p_k) \geq c_1 \|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right\} \tag{6.13}$$

**Theorem**

# Reduction Obtained by The Cauchy Point

**Lemma**

The dogleg and two dimensional subspace minimization algorithms produce approximate solution $p_k$ of the subproblem (3) that satisfy the following estimate of decrease in the model function:

$$m_k(0) - m_k(p_k) \geq c_1 \|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right\} \tag{6.13}$$

**Theorem**

- Let $p_k$ be any vector such that $\|p_k\| \leq \Delta_k$ and $m_k(0) - m_k(p) \geq c_2\left(m_k(0) - m_k(p_k^C)\right)$.

# Reduction Obtained by The Cauchy Point

**Lemma**

The dogleg and two dimensional subspace minimization algorithms produce approximate solution $p_k$ of the subproblem (3) that satisfy the following estimate of decrease in the model function:

$$m_k(0) - m_k(p_k) \geq c_1 \|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right\} \tag{6.13}$$

**Theorem**

- Let $p_k$ be any vector such that $\|p_k\| \leq \Delta_k$ and $m_k(0) - m_k(p) \geq c_2\big(m_k(0) - m_k(p_k^C)\big)$.
- Then $p_k$ satisfies (6.13) with $c_1 = \frac{c_2}{2}$.

# Reduction Obtained by The Cauchy Point

## Lemma

The dogleg and two dimensional subspace minimization algorithms produce approximate solution $p_k$ of the subproblem (3) that satisfy the following estimate of decrease in the model function:

$$m_k(0) - m_k(p_k) \geq c_1 \|g_k\| \min\left\{\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right\} \tag{6.13}$$

## Theorem

- Let $p_k$ be any vector such that $\|p_k\| \leq \Delta_k$ and $m_k(0) - m_k(p) \geq c_2\left(m_k(0) - m_k(p_k^C)\right)$.
- Then $p_k$ satisfies (6.13) with $c_1 = \frac{c_2}{2}$. In practice, if $p_k$ is the exact solution $p^*$ of (6.3), then it satisfies (6.13) with $c_1 = \frac{1}{2}$.

# Convergence to Stationary Points

# Convergence to Stationary Points

- Global convergence results for trust-region methods come in two varieties, depending on whether we set the parameter $\eta$ in Algorithm 4 to zero or to some small positive value.

# Convergence to Stationary Points

- Global convergence results for trust-region methods come in two varieties, depending on whether we set the parameter $\eta$ in Algorithm 4 to zero or to some small positive value.
- when $\eta = 0$ (that is, the step is taken whenever it products a lower value of $f$), we can show that the sequence of gradients $\{g_k\}$ has a limit point at zero.

# Convergence to Stationary Points

- Global convergence results for trust-region methods come in two varieties, depending on whether we set the parameter $\eta$ in Algorithm 4 to zero or to some small positive value.
- when $\eta = 0$ (that is, the step is taken whenever it products a lower value of $f$), we can show that the sequence of gradients $\{g_k\}$ has a limit point at zero.
- For the more stringent acceptance test with $\eta > 0$, which requires the actual decrease in $f$ to be at least some small fraction of the predicted decrease, we have the stronger result that $g_k \to 0$.

We provide the global convergence results for both case.

# Convergence to Stationary Points

We assume throughout that the approximate Hessians $B_k$ are bounded in norm, and that $f$ is bounded below on the level set

$$S \equiv \left\{x | f(x) \leq f(x_0)\right\}. \tag{6.14}$$

For later reference, we define an open neighborhood of this set by

$$S(R_0) \equiv \left\{x | \|x - y\| < R_0 \text{ for some } y \in S\right\}.$$

where $R_0$ is a positive constant.

# Convergence to Stationary Points

We assume throughout that the approximate Hessians $B_k$ are bounded in norm, and that $f$ is bounded below on the level set

$$S \equiv \{x | f(x) \le f(x_0)\}. \tag{6.14}$$

For later reference, we define an open neighborhood of this set by

$$S(R_0) \equiv \{x | \|x - y\| < R_0 \text{ for some } y \in S\}.$$

where $R_0$ is a positive constant.
To allow our results to be applied more generally, we also allow the length of the approximate solution $p_k$ of (6.3) to exceed the trust-region bound, provided that it stays within some fixed multiple of the bound; that is,

$$\|p_k\| \le \gamma \Delta_k, \text{ for some constant } \gamma \le 1. \tag{6.15}$$

# Convergence to Stationary Points

The first result deals with the case $\eta = 0$.

**Theorem**

- Let $\eta = 0$ in Algorithm 4.

# Convergence to Stationary Points

The first result deals with the case $\eta = 0$.

**Theorem**

- Let $\eta = 0$ in Algorithm 4.
- Suppose that $\|B_k\| \leq \beta$ for some constant $\beta$, that $f$ is bounded below on the level set $S$ defined by (6.14) and Lipschitz continuously differentiable in the neighborhood $S(R_0)$ for some $R_0 > 0$, and that all approximate solution of (6.3) satisfy the inequalities (6.13) and (6.15), for some positive constant $c_1$ and $\gamma$.

# Convergence to Stationary Points

The first result deals with the case $\eta = 0$.

**Theorem**

- Let $\eta = 0$ in Algorithm 4.
- Suppose that $\|B_k\| \leq \beta$ for some constant $\beta$, that $f$ is bounded below on the level set $S$ defined by (6.14) and Lipschitz continuously differentiable in the neighborhood $S(R_0)$ for some $R_0 > 0$, and that all approximate solution of (6.3) satisfy the inequalities (6.13) and (6.15), for some positive constant $c_1$ and $\gamma$.
- We then have

$$\lim_{k\to\infty}\inf \|g_k\| = 0. \tag{6.16}$$

# Convergence to Stationary Points

**Theorem (Schultz, Schnabel and Byrd)**

- Let $\eta \in (0, \frac{1}{4})$ in Algorithm 4.

# Convergence to Stationary Points

**Theorem (Schultz, Schnabel and Byrd)**

- Let $\eta \in (0, \frac{1}{4})$ in Algorithm 4.
- Suppose that $\|B_k\| \leq \beta$ for some constant $\beta$, that $f$ is bounded below on the level set $S$ defined by (6.14) and Lipschitz continuously differentiable in the neighborhood $S(R_0)$ for some $R_0 > 0$, and that all approximate solution of (6.3) satisfy the inequalities (6.13) and (6.15), for some positive constant $c_1$ and $\gamma$.

# Convergence to Stationary Points

**Theorem (Schultz, Schnabel and Byrd)**

- Let $\eta \in (0, \frac{1}{4})$ in Algorithm 4.
- Suppose that $\|B_k\| \leq \beta$ for some constant $\beta$, that $f$ is bounded below on the level set $S$ defined by (6.14) and Lipschitz continuously differentiable in the neighborhood $S(R_0)$ for some $R_0 > 0$, and that all approximate solution of (6.3) satisfy the inequalities (6.13) and (6.15), for some positive constant $c_1$ and $\gamma$.
- We then have

$$\lim_{k\to\infty} g_k = 0. \tag{6.17}$$

# Convergence to Stationary Points

- Suppose that the assumptions of above theorem are satisfied and in addition that $f$ is twice continuously differentiable in the level set $S$.

# Convergence to Stationary Points

- Suppose that the assumptions of above theorem are satisfied and in addition that $f$ is twice continuously differentiable in the level set $S$.
- Suppose that $B_k = \nabla^2 f(x_k)$ for all $k$, and that the approximate solution $p_k$ of (6.3) at each iteration satisfies

$$m(0) - m(p) \geq c_1\big(m(0) - m(p^*)\big), \tag{6.18a}$$

$$\|p\| \leq \gamma\Delta, \tag{6.18b}$$

for some fixed $\gamma > 0$. Then

$$\lim_{k\to\infty} \|g_k\| = 0.$$

# Convergence to Stationary Points

- Suppose that the assumptions of above theorem are satisfied and in addition that $f$ is twice continuously differentiable in the level set $S$.
- Suppose that $B_k = \nabla^2 f(x_k)$ for all $k$, and that the approximate solution $p_k$ of (6.3) at each iteration satisfies

$$m(0) - m(p) \geq c_1 \big(m(0) - m(p^*)\big), \tag{6.18a}$$

$$\|p\| \leq \gamma\Delta, \tag{6.18b}$$

  for some fixed $\gamma > 0$. Then

$$\lim_{k\to\infty} \|g_k\| = 0.$$

- In addition, if the level set $S$ of (6.14) is compact, then either the algorithm terminates at a point $x_k$ at which the second-order necessary conditions for a local solution hold, or else $\{x_k\}$ has a limit point $x^*$ in $S$ at which the second-order necessary conditions hold.

# Local Convergence of TR Newton Methods

Theorem

# Local Convergence of TR Newton Methods

### Theorem

- Let $f$ be twice Lipschitz continuously differentiable in a neighborhood of a point $x^*$ at which second-order sufficient conditions are satisfied.

# Local Convergence of TR Newton Methods

## Theorem

- Let $f$ be twice Lipschitz continuously differentiable in a neighborhood of a point $x^*$ at which second-order sufficient conditions are satisfied.
- Suppose the sequence $\{x_k\}$ converges to $x^*$ and that for all $k$ sufficiently large, the trust-region algorithm based on (6.3) with $B_k = \nabla^2 f(x_k)$ chooses steps $p_k$ that satisfy the Cauchy-point-based model reduction criterion (6.13) and are asymptotically similar to Newton steps $p_k^N$ whenever $\|p_k^N\| \leq \frac{1}{2}\Delta_k$, that is,

$$\|p_k - p_k^N\| = o(\|p_k^N\|) \tag{6.19}$$

# Local Convergence of TR Newton Methods

### Theorem

- Let $f$ be twice Lipschitz continuously differentiable in a neighborhood of a point $x^*$ at which second-order sufficient conditions are satisfied.
- Suppose the sequence $\{x_k\}$ converges to $x^*$ and that for all $k$ sufficiently large, the trust-region algorithm based on (6.3) with $B_k = \nabla^2 f(x_k)$ chooses steps $p_k$ that satisfy the Cauchy-point-based model reduction criterion (6.13) and are asymptotically similar to Newton steps $p_k^N$ whenever $\|p_k^N\| \le \frac{1}{2}\Delta_k$, that is,

$$\|p_k - p_k^N\| = o(\|p_k^N\|) \tag{6.19}$$

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

It is immediate from the above theorem that if $p_k = p_k^N$ for all $k$ sufficiently large, we have quadratic convergence of $\{x_k\}$ to $x^*$.

# Scaling

- Recalling our definition of a trust region

# Scaling

- Recalling our definition of a trust region - a region around the current the current iterate within which the model $m_k(\cdot)$ is an adequate representation of the true objective $f(\cdot)$

# Scaling

- Recalling our definition of a trust region - a region around the current the current iterate within which the model $m_k(\cdot)$ is an adequate representation of the true objective $f(\cdot)$ - it is easy to see that a *spherical* trust region may not be approximated when $f$ is poorly scaled.

# Scaling

- Recalling our definition of a trust region - a region around the current the current iterate within which the model $m_k(\cdot)$ is an adequate representation of the true objective $f(\cdot)$ - it is easy to see that a *spherical* trust region may not be approximated when $f$ is poorly scaled.
- Even if the model Hessian $B_k$ is exact, the rapid changes in f along certain directions probably will cause $m_k$ to be a poor approximation to $f$ along these directions.

# Scaling

- Recalling our definition of a trust region - a region around the current the current iterate within which the model $m_k(\cdot)$ is an adequate representation of the true objective $f(\cdot)$ - it is easy to see that a *spherical* trust region may not be approximated when $f$ is poorly scaled.
- Even if the model Hessian $B_k$ is exact, the rapid changes in f along certain directions probably will cause $m_k$ to be a poor approximation to $f$ along these directions.
- On the other hand, $m_k$ may be a more reliable approximation to $f$ along these directions in which $f$ is changing more slowly.

Since the shape of the trust region should be such that the confidence in the model is more or less the same at all points on the boundary of the region, we are led naturally to consider *elliptical* trust regions in which the axes are short in the sensitive directions and longer in the less sensitive directions.

# Scaling

# Scaling

- Elliptical trust regions can be defined by

$$\|Dp\| \le \Delta, \tag{6.20}$$

where $D$ is a diagonal matrix with positive diagonal elements, yielding the following scaled trust-region subproblem:

$$\lim_{p\in\mathcal{R}^n} m_k(p) \equiv f(x_k) + \nabla f(x_k)^T p + \frac{1}{2}p^T B_k p, \quad s.t. \quad \|Dp\| \le \Delta_k. \tag{6.21}$$

# Scaling

- Elliptical trust regions can be defined by

$$\|Dp\| \leq \Delta, \tag{6.20}$$

where $D$ is a diagonal matrix with positive diagonal elements, yielding the following scaled trust-region subproblem:

$$\lim_{p \in \mathcal{R}^n} m_k(p) \equiv f(x_k) + \nabla f(x_k)^T p + \frac{1}{2} p^T B_k p, \quad s.t. \quad \|Dp\| \leq \Delta_k. \tag{6.21}$$

- When $f(x)$ is highly sensitive to the value of the $i$th component $x_i$ , we set the corresponding diagonal element $d_{ii}$ of $D$ to be large, while $d_{ii}$ is smaller for less-sensitive components.

# Trust Region in Other Norms

- Trust regions may also be defined in terms of norms other than the Euclidean norm.

# Trust Region in Other Norms

- Trust regions may also be defined in terms of norms other than the Euclidean norm.
- For instance, we may have

$$\|p\|_1 \leq \Delta_k, \text{ or } \|p\|_\infty \leq \Delta_k,$$

# Trust Region in Other Norms

- Trust regions may also be defined in terms of norms other than the Euclidean norm.
- For instance, we may have

$$\|p\|_1 \leq \Delta_k, \text{ or } \|p\|_\infty \leq \Delta_k,$$

- or their scaled counterparts

$$\|Dp\|_1 \leq \Delta_k, \text{ or } \|Dp\|_\infty \leq \Delta_k.$$

Thanks For Your Attention