6

UNCONSTRAINED NONLINEAR OPTIMIZATION

6.1 Optimality conditions

common notations

- x_k: kth iteration point of x.
- g_k: gradient $\nabla f(x_k)$ at x_k.
- H_k: Hessian $\nabla^2 f(x_k)$ at x_k.

6.1.1 Optimality concepts

In **unconstrained nonlinear optimization**, the goal is to generally minimize an objective function $f(x) : \mathbb{R}^N \to \mathbb{R}$ over $\mathbb{R}^N$. $f(x)$ is required to be in C_1(sometimes C_2 is required).

Example 6.1.1. Examples of objective functions include

- Linear functions such as $f(x) = cx, c \in \mathbb{R}$.
- Quadratic functions such as $f(x_1, x_2) = x_1^2 + 2x_1x_2 + 4x_2^2$.
- Polynomial functions such as $f(x) = a_0 + a_1x + a_2x_2 + \cdots a_px_p$.
- Complex nonlinear function such as the Humpback function [Figure 6.1.2] given by

$$f(x_1, x_2) = x_1^2(4 - 2.1x_1^2 + 0.33x^4) + x_1x_2 + x_2^2(-4 + 4x_2).$$

In searching for solutions to the optimization problems, we usually distinguish local and global minimums.

Definition 6.1.1 (local and global minimum). *[1, p. 5] A vector x^* is an **unconstrained local minimum** of a function $f : \mathbb{R}^n \to \mathbb{R}$ if it is no worse than its neighbors; that is, if there exist an $\delta > 0$ such that*

$$f(x) \geq f(x^*), \forall \left\| x - x^* \right\| < \delta$$

*A vector x^* is an **unconstrained global minimum** of f if $f(x^*) \geq f(x), \forall x \in \mathbb{R}^n$.*

Remark 6.1.1 (possible non-existence of minimizer). For some $f(x)$, the minimizer might not exist. For example, linear uncontrained optimization like $f(x) = x$ does not have a minimum. **In nature, this is because $\mathbb{R}^N$ is not a compact set.**

Local and global minimums can be further categorized into more specific types [Figure 6.1.1]:

Definition 6.1.2 (types of local minimizer). *Let a function $f(x)$ be defined in a region $D \subset \mathbb{R}^n$. We say that $x^* \in D$ is a*

- ***a global minimizer*** *of $f(x)$ in D if $f(x^*) \leq f(x)$ for all $x \in D$.*
- ***a strict global minimizer*** *of $f(x)$ in D if $f(x^*) < f(x)$ for all $x \in D, x \neq x^*$.*
- ***a local minimizer*** *of $f(x)$ in D if $f(x^*) \leq f(x)$ for all x in some open ball $B(x^*)$.*
- ***a strict local minimizer*** *of $f(x)$ in D if $f(x^*) < f(x)$ for all x in some open ball $B(x^*), x \neq x^*$.*

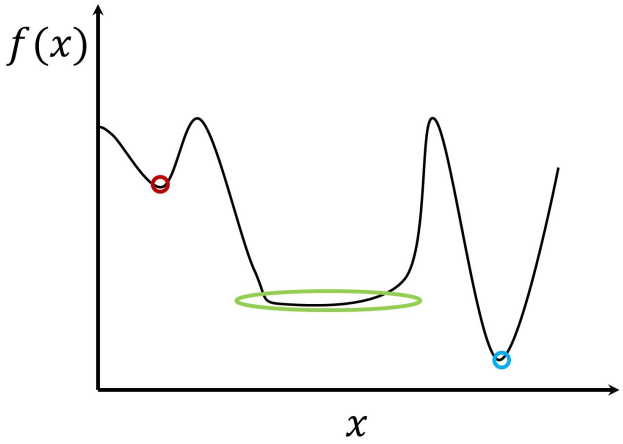


Figure 6.1.1: Demonstration of local minimizer (red, green, and blue), strict lcoal minimizer (red and blue), and global minimizer (blue).

In practice, the objective function can be a highly complex nonlinear function [Figure 6.1.2]. Searching for global minimum can be intractable. In the following, we will develop theories and algorithms can allow us to find one local minimum.

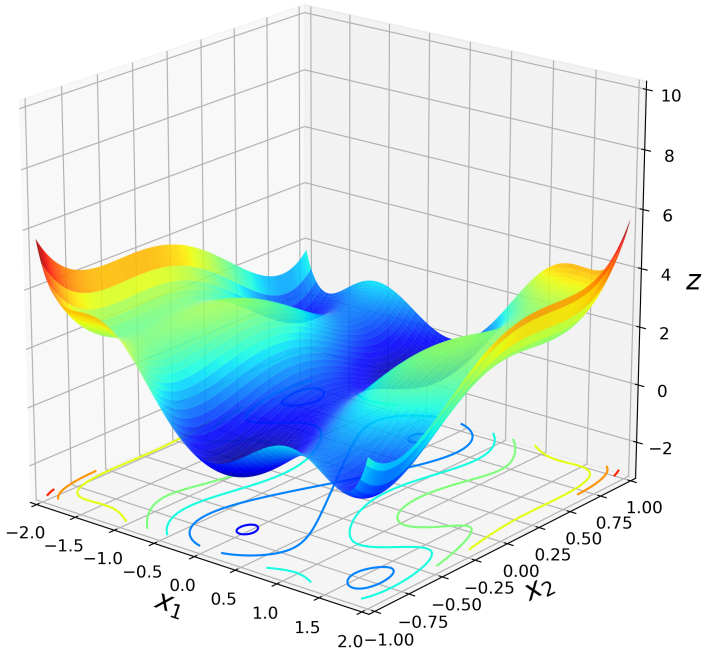


Figure 6.1.2: A complex objective function in unconstrained optimization.

6.1.2 Necessary and sufficient conditions

Theorem 6.1.1 (first order necessary condition). *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable, if x^* is local minimizer, then $\nabla f = 0$.*

Proof. Use contradiction to prove, if $g = \nabla f \neq 0$, and assume $\nabla f > 0$, then we can show

$$f(x^* - ag(x^*)) = f(x^*) - a\|g(x^*)\|^2 + O(a^2).$$

Then for sufficiently small a, we have

$$f(x^* - ag(x^*)) - f(x^*) < 0,$$

which contradicts with the fact that $f(x^*)$ is local minimum. □

Remark 6.1.2. This is not a sufficient condition. Consider a counter example is $f(x) = x^3$. At $x = 0$, $f'(0) = 0$ but it is not a local minimum.

Theorem 6.1.2 (second order necessary condition). *Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable, if x^* is local minimizer, then $\nabla^2 f(x^*) = H(x^*)$ is **positive semidefinite**, i.e.,*

$$s^T H(x^*)s \geq 0, \forall s \in \mathbb{R}^n.$$

Proof. If x^* is local minimizer, from the first order necessary condition above, we know $\nabla f(x^*) = 0$. Suppose there exist a direction s such that $s^T H(x^*)s < 0$, then the Taylor expansion along s will give

$$f(x^* + as) = f(x^*) + \frac{1}{2}a^2 s^T H(x^*)s + O(a^3).$$

We select a sufficiently small $a > 0$ such that

$$\frac{1}{2}a^2 s^T H(x^*)s + O(a^3) < 0$$

Then, we have

$$f(x^* + as) - f(x^*) < 0,$$

which contradicts with the fact that $f(x^*)$ is local minimum. □

Theorem 6.1.3 (second order sufficient optimality condition, strict local minimizer). *[1] Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable, if $\nabla f(x^*) = 0$ and $H = \nabla^2 f(x^*)$ is **positive definite**, then x^* is local minimizer.*

In fact, if the above conditions holds, it can be showed that there exists scalars $\gamma > 0$ and $\epsilon > 0$ such that

$$f(x) \geq f(x^*) + \frac{\gamma}{2}\|x - x^*\|, \forall x \text{ such that } \|x - x^*\| < \epsilon.$$

Proof. Consider an open Ball $B(x^*, \epsilon)$ for some ϵ. For any $s \neq 0, x^* + s \in B(x^*, \epsilon)$, we have

$$\begin{aligned} f(x^* + s) &= f(x^*) + g(x^*)^T s + \frac{1}{2}s^T H(x^*)s + o(\|s\|^2) \\ &\geq f(x^*) + \frac{1}{2}\lambda\|s\|^2 + o(\|s\|^2) \\ &= f(x^*) + \frac{1}{2}\|s\|^2 (\lambda + \frac{o(\|s\|^2)}{\|s\|^2}) \\ &> f(x^*), \end{aligned}$$

for sufficiently small ϵ. Note that $\lambda > 0$ is the smallest eigenvalue of $H(x^*)$, and we use Rayleigh quotient [Theorem 4.8.4]that

$$s^T H s \geq \lambda \|s\|^2 .$$

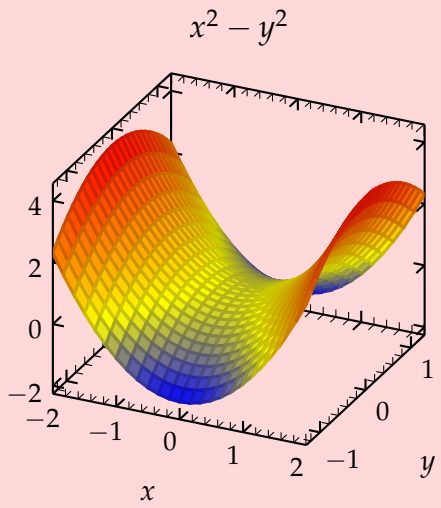
□

Remark 6.1.3 (lack of first-order sufficient condition). Note that **there is no first-order sufficient condition**, i.e., any first-order condition cannot guarantee sufficiency unless we know the function is convex [Theorem 9.5.3].

Example 6.1.2.

- $f(x) = x^2, f'(x) = 2x = 0, x^* = 0, f''(x^*) = 2 > 0$. Therefore, $x = 0$ is a local minimum.
- $f(x) = x^3, f'(x) = 3x^2 = 0, x^* = 0, f''(x^*) = 0$. Therefore, $x = 0$ is not a local minimum or maximum.
- $f(x) = x^4, f'(x) = 4x^3 = 0, x^* = 0, f''(x^*) = 0$. Note that optimal conditions we cover so far can be determine if $x = 0$ is optimal or not. We need to look for higher order conditions.

Example 6.1.3. Consider the function $f(x, y) = x^2 - y^2$.



Note that at $(0,0)$, gradient is zero, but its hessian $H = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$ is not semi-positive definite. Therefore $(0,0)$ is not a local minimum.

6.1.3 Special case: unconstrained quadratic programming

Theorem 6.1.4 (optimality for unconstrained quadratic programming). *Let $f(x) = c^T x + \frac{1}{2} x^T H x$, where H is symmetric and $c \neq 0$.*

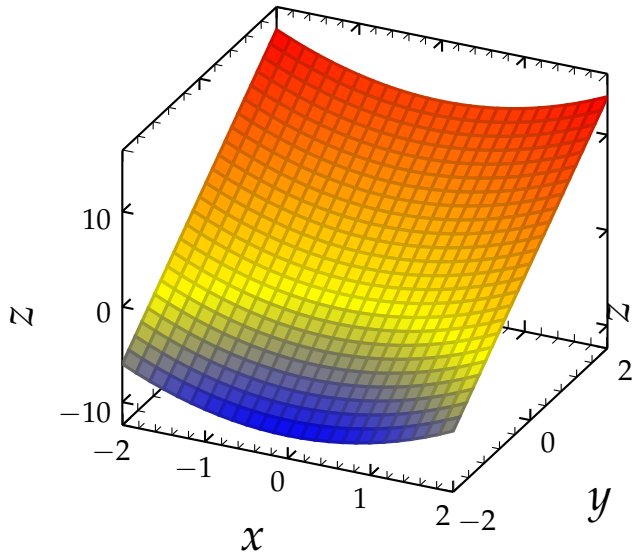
We have the following [also see Figure 6.1.3]

- *$f(x)$ is unbounded below if any the three conditions holds*
  - *$H = 0$.*
  - *$c \notin \mathcal{R}(H)$; that is, equation $Hx^* = -c$ has no solution.*
  - *H has negative eigenvalues.*
- *$f(x)$ has a unique global minimizer $x*$ if $Hx^* = -c$ has a unique solution and H is positive definite.*
- *$f(x)$ has an infinitely many global minimizers $x*$ if $Hx^* = c$ has infinitely many solutions and H is positive semidefinite.*

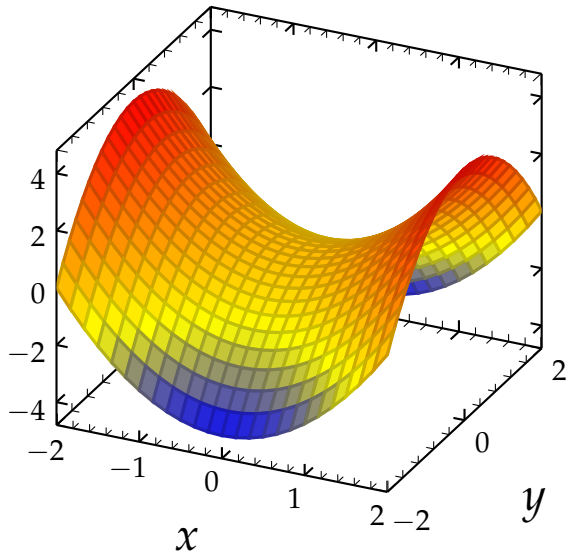
Proof. (1) (a) straight forward. (b) If $Hx^* = c$ has no solution, then c must has component c_N lying in the null space of H. Move along c_N and $f(x)$ can reach ∞. (c) Move along the direction associated with negative eigenvalue and $f(x)$ can reach ∞. (2) From (3) Consider a perturbed point $x^* + p$, we have

$$\begin{aligned} &\frac{1}{2}(x^* + p)^T H(x^* + p) + c^T(x^* + p) - \frac{1}{2}[x^*]^T H x^* - c^T x^* \\ =&\frac{1}{2} p^T H p + c^T p + p^T H x \\ =&\frac{1}{2} p^T H p \geq 0 \end{aligned}$$

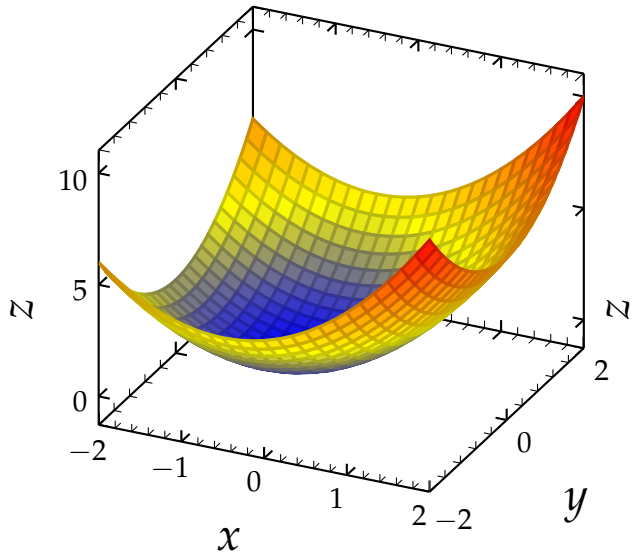
In fact, if $p \in \mathcal{N}(H)$, then $x^* + p$ is also a global minimizer. □



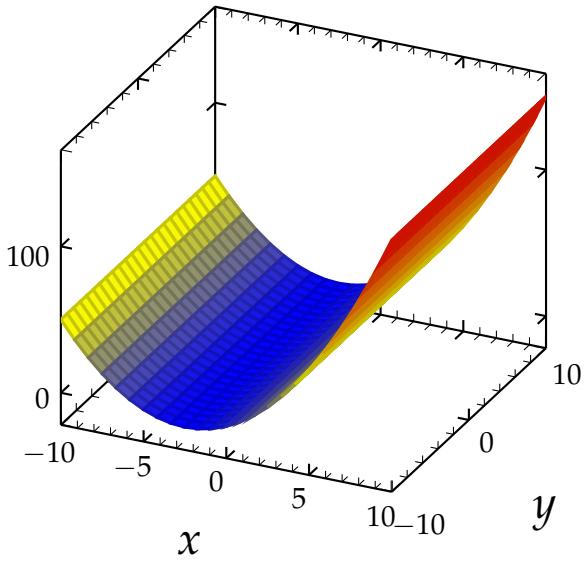
(a) $f(x,y) = 5y + x^2$ where $Hx^* = c$ does not have a solution.



(b) $f(x,y) = x^2 - y^2$ where H has a negative eigenvalue.



(c) $f(x,y) = x^2 + y^2 + x$ where H is positive definite.



(d) $f(x,y) = 5x + x^2$ where H is positive semi-definite and $Hx^* = -c$ has infinitely many solutions.

Figure 6.1.3: Illustration of different cases in unconstrained quadratic optimization.

6.2 Line search method

common notations

- x_k: kth iteration point of x.
- g_k: gradient $\nabla f(x_k)$ at x_k.
- H_k: Hessian $\nabla^2 f(x_k)$ at x_k.

6.2.1 A generic algorithm

Given a differentiable function $f(x)$ and its non-zero gradient[1] $g(x) \triangle \nabla f(x)$, if we follow the direction $-g$ move a sufficiently-small step size α, we can decrease $f(x)$, i.e.,

$$f(x - \alpha g(x)) < f(x)$$

using Taylor expansion.

The direction $-g(x)$ is the steepest descent direction. More generally, we can select a descent direction of similar nature, perform a suitable step size and decrease the $f(x)$. If perform aforementioned steps iteratively, we could constantly decrease $f(x)$ until a minimum is reached.

This is indeed the core idea of **line search** framework. In summary, we are given an objective function $f(x) \in C_1$ (sometimes C_2 is required). Starting from an initial iterate x_0, we repeat the following procedure

- **(find a descent direction)**find a descent direction p_k .
- **(choose step length)** compute a scalar step length α_k such that $f(\alpha_k p_k + x_k) < f(x_k)$
- Generate the next iteration via

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

The procedure can also be summarized by the following algorithm 1.

Algorithm 1: A generic line search algorithm

Input: Initial guess x_0

Set $k = 0$ **repeat**

  Find a decent direction p_k at x_k.

  Compute a suitable step length α_k such that $f(\alpha_k p_k + x_k) < f(x_k)$

  Set $x_{k+1} = x_k + \alpha_k p_k$

  Set $k = k + 1$

until *terminal condition is satisfied*;

Output: approximate minimizer x_k

1 If gradient is zero, we have possibly arrived the local minimum, see Theorem 6.1.1

In the following sections, we will elaborate the theory and algorithm regarding the computation of descent directions and step size. By combining different descent direction and step size calculation subroutines, we get different line search algorithms.

6.2.2 Theory and computation of descent directions

6.2.2.1 *Gradient descent direction and properties*

Definition 6.2.1 (descent direction). *A vector $p_k \in \mathbb{R}^N$ at x_k is called a descent direction if*

$$g_k^T p_k < 0$$

if $g^T \neq 0$.

Remark 6.2.1. We are not considering the situation where $g_k = 0$, which suggesting local minimum has arrived(and the iteration process should be terminated.).

Lemma 6.2.1. *Let p_k be a descent direction at x_k. Then there exists a $\alpha > 0$ such that*

$$f(x_k - \alpha p_k) - f(x_k) < 0.$$

Proof. We can write the descent direction as the combination of $-g_k$ and a component perpendicular to g_k. That is, we have

$$p_k = -\beta g_k + \gamma g_k^{\perp}, \beta > 0.$$

Using Taylor expansion, we can show

$$f(x_k - \alpha p_k) = f(x_k) - \alpha\beta \left\| g_k \right\|^2 + O(\alpha^2).$$

Then for sufficiently small α, we have

$$f(x_k - \alpha\beta g_k) - f(x_k) < 0.$$

□

Lemma 6.2.2 (steepest decent direction and their properties). *Let x_k be the current iterate. Let g_k be the gradient at x_k.*

- *Let g_k be the gradient at x_k, then $p_k = -g_k$ is a descent direction, and is called **steepest-descent direction**.*

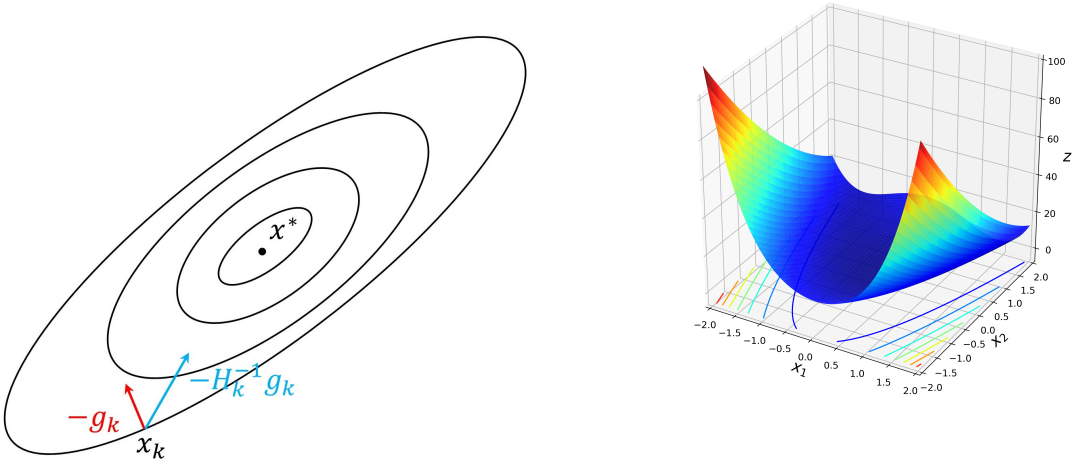
- *The steepest descent direction $p_k = -g_k$ is the solution to a constrained minimization based on a linear model*

$$\min_{p\in\mathbb{R}^n} m_k^L(x_k + p) = f_k + g_k^T p, \text{ subject to } \|p\|_2 = \|g_k\|.$$

- *The steepest descent direction $p_k = -g_k$ is the solution to a strict convex quadratic programming*

$$\min_{p\in\mathbb{R}^n} f_k + g_k^T p + \frac{1}{2} p^T I p.$$

Proof. (1) $g_k^T p_k = -\|g_k\|^2 < 0$ if $g_k \neq 0$. (2) The Lagrangian is $L(p,\lambda) = f_k + g_k^T p + \lambda(\frac{1}{2}p^T p - g_k^T g_k)$. The optimality conditions gives $g_k + \lambda p = 0, \|p\|_2 = \|g_k\|$; that is, $p = -g_k$ or g_k. It is easy to see that minimizer is $-g_k$. (3) Straight forward. □



(a) Comparison of steepest descent direction and Newton step direction.

(b) Steepest gradient descent would perform poorly for the RosenBrock objective function, which contains long and shallow valley.

Figure 6.2.1: Drawbacks of steepest gradient descent.

6.2.2.2 *Curvature-modified descent direction*

Lemma 6.2.3 (modified decent directions). *Let x_k be the current iterate. Let g_k be the gradient at x_k.*

- *Let B_k be a symmetric positive definite matrix, then the p_k is a descent direction if $B_k p_k = -g_k$.*
- *The descent direction p_k as the solution $B_k p_k = -g_k, B_k > 0$ is the solution to a strict convex quadratic programming*

$$\min_{p \in \mathbb{R}^n} f_k + g_k^T p + \frac{1}{2} p^T B_k p.$$

Proof. (1) $g_k^T p_k = -g_k B_k^{-1} g_k < 0$ if $g_k \neq 0$ since B_k^{-1} is also symmetric positive definite [Theorem 4.7.3]. (2) The optimality condition gives $g_k + B_k p = 0 \implies B_k p_k = -g_k$. □

Definition 6.2.2 (Newton direction). *Let g_k and H_k be the gradient and the **Hessian** at x_k, then the p_k satisfying $H_k p_k = -g_k$ is called **Newton direction**.*

Lemma 6.2.4 (properties of Newton direction).

- *Newton's direction is a **descent direction** if $H_k > 0$.*
- *Let p be the Newton direction and let g_k be the gradient at x_k. It follows that p is the minimizer of*

$$\min_{p \in \mathbb{R}^n} \frac{1}{2} p^T H_k p + g_k^T x,$$

if $H_k > 0$.

Proof. (1) $g^T p = -g^T H^{-1} g < 0$.(2) Consider the objective function

$$\frac{1}{2} p^T H_k p + g_k^T x.$$

The optimality condition is

$$H_k p + g_k = 0 \implies H_k p = -g_k.$$

That is p is also the Newton direction. □

Example 6.2.1. Consider the optimization

$$\min(x_1, x_2) = (x_1 - 2)^4 + (x_1 - 3x_2)^2 + \exp(2x_1 - 2).$$

The gradient is given by

$$g = \nabla f(x_1, x_2) = \begin{bmatrix} 4(x_1-2)^3 + 2(x_1 - 3x_2) + 2\exp(2x_1 - 2) \\ -4(x_1 - 3x_2) \end{bmatrix},$$

and Hessian is given by

$$H = \nabla^2 f(x_1, x_2) = \begin{bmatrix} 12(x_1-2)^2 + 2 + 4\exp(2x_1 - 2) & -6 \\ -6 & 18 \end{bmatrix}.$$

The Newton direction can be evaluated by

$$p_{NW} = -H^{-1} \cdot g.$$

If the Hessian H_k is not positive definite (H_k will always be symmetric), then the Newton direction is not a descent direction. In the following, we provide a way to generate a positive definite matrix B_k from H_k, that also at the same time contains some curative information.

Methodology 6.2.1 (Hessian modification method via eigenvalue spectrum). *Given a symmetric matrix H with decomposition $H = V\Lambda V^T$, we can use the following procedures to generate a positive definite matrix B:*

- *Make all eigenvalue $\lambda_i' = \epsilon > 0$ if $\lambda_i < \epsilon$; otherwise $\lambda_i' = \lambda_i$. And the modified matrix is $B = V\Lambda' V^T$.*
- *Make all eigenvalue $\lambda_i' = -\lambda_i$ if $\lambda_i < -\epsilon$; $\lambda_i' = \lambda_i$ if $\lambda_i' > \epsilon$; $\lambda_i' = \epsilon$ otherwise. And the modified matrix is $H' = V\Lambda' V^T$. And the modified matrix is $B = V\Lambda' V^T$.*
- *Directly generate $B = H + (\min\{\lambda_i\} + \epsilon)I$.*

6.2.2.3 *Quasi-Newton method*

In Methodology 6.2.1, we introduced a rather Naive way to modify the spectrum Hessian H, which contains curative information, to a positive definite matrix B, in order to generate descent direction p satisfying $Bp = -g$.

This section, we discuss an alternative way, known as Quasi-Newton method, to generate B that approximate H. In the Quasi-Newton method, we aims to efficiently compute B_{k+1} that satisfies

$$B_{k+1}(x_{k+1} - x_k) = \nabla f(x_{k+1}) - \nabla f(x_k)$$

which is known as **Secant equation**. Intuitively, Secant equation is aiming to solve Hessian via finite different. If x is one-dimension, we recover the classical root finding Secant method [Methodology A.14.3].

Denote $s_k = (x_{k+1} - x_k)$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$,the secant equation is also written as $B_{k+1}s_k = y_k$.

Note that Secant equation is an underdetermined system for B. There are many Quasi-Newton algorithms [2, p. 135], which give different ways to compute B that satisfies Secant equation.

Here, we cover the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, which solve B_{k+1} on top of B_k via

$$B_{k+1} = B_k + \alpha uu^T + \beta vv^T$$

with $u = y_k$ and $v = B_k s_k$

By requiring the satisfactions of the Secant equation, we have

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k}$$

Using matrix inversion formula [Lemma A.8.3], we can get

$$B_{k+1}^{-1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) B_k^{-1} \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k}.$$

The BFGS gives a positive definite B_{k+1} if certain condition is satisfied, as given in the following.

Lemma 6.2.5 (Positive definiteness of BFFS update). *Let B_k be positive definite. Then*

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k}$$

is posiitive definite if $s_k^T y_k > 0$.

Proof.

$$\begin{aligned} x^T B_{k+1} x &= x^T B_k x + \frac{\left(s_k^T x\right)^2}{s_k^T y_k} - \frac{\left(y_k^T B_k x\right)^2}{y_k^T B_k y_k} \\ &= \frac{y_k^T B_k y_k x^T B_k x - \left(y_k^T B_k x\right)^2}{y_k^T B_k y_k} + \frac{\left(s_k^T x\right)^2}{s_k^T y_k} \end{aligned}$$

If one defines the dot product $\langle x, y \rangle$ as $x^T B_k y$, the above equation reads

$$x^T B_{k+1} x = \frac{\langle y_k, y_k \rangle \langle x, x \rangle - \langle y_k, x \rangle^2}{\langle y_k, y_k \rangle} + \frac{\left(s_k^T x\right)^2}{s_k^T y_k}$$

where the first term is positive due to Cauchy-Schwartz inequality, and the second is positive since $s_k^T y_k > 0$. □

As we will see later, the condition $s_k^T y_k > 0$ is satisfied if the step size is chosen according to the Wolfe condition [Definition 6.2.4].

6.2.2.4 *Subspace optimization in quadratic forms*

In this section, we study a simple quadratic optimization problem

$$f(x) = \frac{1}{2} x^T H x.$$

We are particularly interested in the relationship between minimization and eigenvalue spectrum. Intuitively, if we want to increase the objective function, we can move away from origin along the positive eigenvector direction. Or we can move closer to the origin along the negative eigenvector direction.

We present the results in the following Lemma.

Lemma 6.2.6. *Consider objective function*

$$f(x) = \frac{1}{2} x^T H x$$

where $x \in \mathbb{R}^N$, and $H \in \mathbb{R}^{N \times N}$ is a symmetric and nonsingular matrix. Let the spectral decomposition of H be $H = V \Lambda V^T$ Therefore, if we want to increase $f(x)$ from an initial point x_0,

- *we can move in the direction $p = V_i(V_i^T x)$ where $\lambda_i > 0$; that is $f(x_0 + \alpha p) > f(x_0)$, if $\alpha > 0$ is sufficiently small.*
- *we can move in the direction $p = -V_j(V_j^T x)$ where $\lambda_j < 0$; that is $f(x_0 + \alpha p) > f(x_0)$, if $\alpha > 0$ is sufficiently small.*

Similarly, if we want to decrease $f(x)$,

- *we can move in the direction $p = -V_i(V_i^T x)$ where $\lambda_i > 0$; that is $f(x_0 + \alpha p) < f(x_0)$, if $\alpha > 0$ is sufficiently small..*

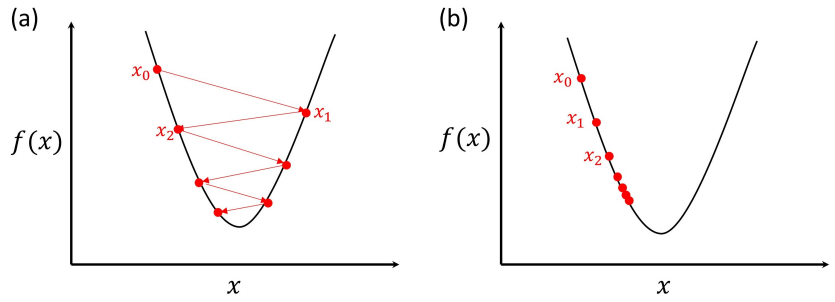


Figure 6.2.2: Demonstration on the step choices on the iterative algorithm. (a) Large step size. (b) Small step size.

- *we can move in the direction* $p = V_j(V_j^T x)$ *where* $\lambda_j < 0$*; that is* $f(x_0 + \alpha p) < f(x_0)$*, if* $\alpha > 0$ *is sufficiently small.*

Proof. $f(x) = \frac{1}{2}x^T Hx = (V^T x)^T \Lambda (V^T x) = \sum_{i=1}^N \lambda_i \frac{1}{2}(V_i^T x)^2$. Then, we can show $\nabla f = \sum_{i=1}^N \lambda_i (V_i^T x) V_i$. And for $\lambda_i > 0$,$(V_i^T x)V_i$ will be the ascent direction. Similar arguments for $\lambda_i < 0$ cases. □

6.2.3 Theory and computation of step length

6.2.3.1 *Overview*

After determining the descent direction to update iterate, we need to determine a suitable step size α to update iterate $x_k = x_{k-1} + \alpha p_k$. The step size α has to be chosen with caution. As shown in Figure 6.2.2, a large step size can result in oscillation of iterates around the local minimum and thus slow down the convergence or even cause divergence. On the other hand, a sufficiently small step size can require extensive iteration to converge, or even fail to converge.

In the following, we first consider some theories for choosing step sizes for well-behaved and relatively simple objective functions. Then we move a more general numerical procedures to determine the step size at each iteration.

6.2.3.2 *Lipschitz bounded convex functions*

Theorem 6.2.1. *Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, and additionally*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| for\ any\ x, y$$

i.e. , ∇f is Lipschitz continuous with constant $L > 0$

Gradient descent with fixed step size $t \leq 1/L$ satisfies

$$f(x^{(k)} - f(x^\star) \leq \frac{\|x^{(0)} - x^\star\|^2}{2tk}$$

i.e. gradient descent has convergence rate $O(1/k)$
i.e. to get $f(x^{(k)}) - f(x^\star) \leq \epsilon$, we need $O(1/\epsilon)$ iterations

Proof. Since ∇f Lipschitz with constant L, which means $\nabla^2 f \preceq LI$, we have $\forall x, y, z$

$$(x - y)^T(\nabla^2 f(z) - LI)(x - y) \leq 0$$

Which means

$$L\|x - y\|^2 \geq (x - y)^T \nabla^2 f(z)(x - y)$$

Based on Taylor's Remainder Theorem, we have $\forall x, y$, $\exists z \in [x, y]$

$$\begin{aligned} f(y) &= f(x) + \nabla f(x)^T(y - x) + \frac{1}{2}(x - y)^T \nabla^2 f(z)(x - y) \\ &\leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|y - x\|^2 \end{aligned} \tag{1}$$

Plugging in $x^+ = x - t\nabla f(x)$,

$$\begin{aligned} f(x^+) &\leq f(x) + \nabla f(x)^T(x - t\nabla x - x) + \frac{L}{2}\|x - t\nabla x - x\|^2 \\ &= f(x) - (1 - \frac{Lt}{2})t\|\nabla f(x)\|^2 \end{aligned} \tag{2}$$

Taking $0 < t \leq 1/L$, $1 - Lt/2 \geq 1/2$, we have

$$f(x^+) \leq f(x) - \frac{t}{2}\|\nabla f(x)\|^2$$

Since f is convex, $f(x) \leq f(x^*) + \nabla f(x)^T(x - x^*)$ we have

$$\begin{aligned} f(x^+) &\leq f(x) - \frac{t}{2}\|\nabla f(x)\|^2 \\ &\leq f(x^*) + \nabla f(x)^T(x - x^*) - \frac{t}{2}\|\nabla f(x)\|^2 \\ &= f(x^*) + \frac{1}{2t}(\|x - x^*\|^2 - \|x - x^* - t\nabla f(x)\|^2) \\ &= f(x^*) + \frac{1}{2t}(\|x - x^*\|^2 - \|x^+ - x^*\|^2) \end{aligned} \tag{3}$$

Summing over iterations, we have

$$\begin{aligned} \sum_{i=1}^{k}(f(x^{(i)} - f(x^*)) &\leq \frac{1}{2t}(\|x^{(0)} - x^*\|^2 - \|x^{(k)} - x^*\|^2) \\ &\leq \frac{1}{2t}\|x^{(0)} - x^*\|^2 \end{aligned} \tag{4}$$

From (2), we can see that $f(x^{(k)})$ is nonincreasing. Then we have

$$f(x^{(k)}) - f(x^*) \leq \frac{1}{k}\sum_{i=1}^{k}(f(x^{(i)} - f(x^*)) \leq \frac{\|x^{(0)} - x^*\|^2}{2tk}$$

□

Lemma 6.2.7 (theoretical optimal step size for unconstrained quadratic optimization). *Consider a unconstrained quadratic minimization problem*

$$\min_{x \in \mathbb{R}^n} f(x) \triangleq \frac{1}{2}x^T H x + b^T x,$$

where $H \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n$. Let $\lambda_1 \geq \lambda_2 ... \geq \lambda_n > 0$ be the eigenvalues of H. It follows that

- *$x_{k+1} = x_k - \alpha \nabla f(x_k)$ can be written as $(x_{k+1} - x^*) = (I - \alpha H)(x_k - x^*)$, where x^* is the unique minimizer of $f(x)$.*
- *If step size $0 < \alpha < 2/\|H\|_2$, then the operator $\|I - \alpha H\|_2 < 1$; that is, $I - \alpha H$ is a contraction.*
- *If $\alpha = 2/(\lambda_1 + \lambda_n)$, then $I - \alpha H$ has the minimum 2-norm.*

Proof. (1) Note that $\nabla f(x_k) = Hx_k - b$ and $Hx^* = b$, therefore

$$x_{k+1} = x_k - \alpha(Hx_k - Hx^*) \Leftrightarrow (x_{k+1} - x^*) = (I - \alpha H)(x_k - x^*)$$

. (2) Note that

$$(x_{k+1} - x^*) = (I - \alpha H)(x_k - x^*)$$

implies [Theorem 4.13.1]

$$\left\|x_{k+1} - x^*\right\| \leq \|I - \alpha H\| \left\|x_k - x^*\right\|.$$

Moreover $\|I - \alpha H\|_2$ equals the maximum **absolute** eigenvalue of $I - \alpha H$ [Theorem 4.9.3], which is $\max|1 - \alpha\lambda_i|$. To let $\|I - \alpha H\|_2 < 1$, we have $\alpha < 2/\lambda_1 = 2/\|H\|_2$. (3) To make $\max|1 - \alpha\lambda_i|$ minimal, we simplify to

$$\min_{\alpha} \max(|1 - \alpha\lambda_1|, |1 - \alpha\lambda_n|)$$

due to the factor that λ_i are monotone. Then, the minimum value is reached at

$$1 - \alpha\lambda_1 = -(1 - \alpha\lambda_n).$$

□

Remark 6.2.2 (convergence rate). At optimal choice of α, we have

$$\|I - \alpha H\|_2 = \frac{\lambda_1/\lambda_n - 1}{\lambda_1/\lambda_n + 1} = \frac{cond(H) - 1}{cond(H) + 1}.$$

We have the following statement about convergence rate (using contraction theory section 5.2]:

- 'fast linear convergence' when $cond(H) \approx 1$.
- 'slow linear convergence' when $cond(H) \gg 1$.
- 'convergence in a single iteration' when $cond(H) = 1$, i.e., for diagonal matrices.

Remark 6.2.3 (compare with conjugate gradient descent). For this type of problem, conjugate gradient descent at most takes n iteration; but using our steepest descent method can take much more longer.

6.2.3.3 *Backtracking-Armijo step side search*

One way to choose step size is to require that the new iterate can lead to an improvement in the function value is at least a fraction of the improvement predicted by the linear approximation [Figure 6.2.3]. Mathematically, the condition is known as **Armijo sufficient decrease condition**, defined by

Definition 6.2.3 (Armijo sufficient decrease condition). *Given a point x_k and a search direction p_k, we say the step size α_k satisfies the **Armijo condition** if*

$$f(x_k + \alpha_k p_k) \leq f(x_k) + \eta \alpha_k g_k^T p_k$$

for some $\eta \in (0,1)$.

To ensure the decrease in f is substantial enough to ensure convergence. When we choose a small η, we tend to select large α_k to ensure the condition to hold.

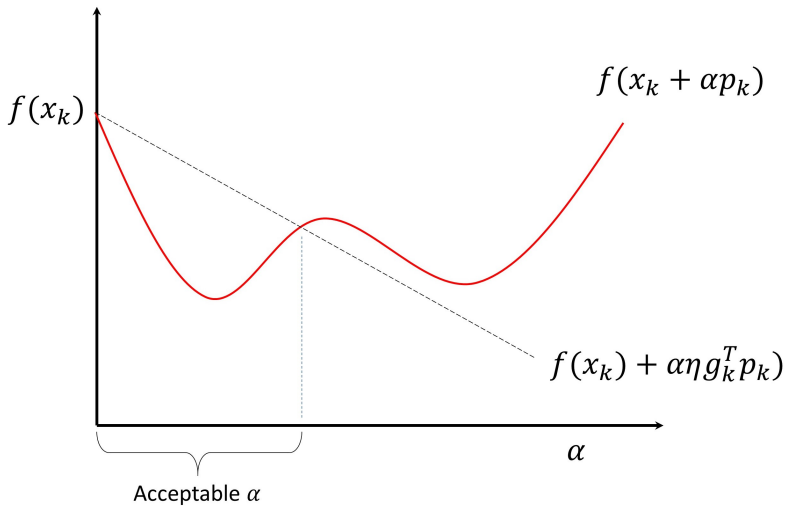


Figure 6.2.3: Armijo sufficient decrease condition.

Algorithm 2: Backtracking-Armijo line search algorithm

Input: Initial guess x_k, p_k
Choose $\alpha_{init} > 0, \eta \in (0,1)$, and $\tau \in (0,1)$
Set $\alpha_0 = \alpha_{init}$ and $l = 0$
Set $l = 0$ **repeat**
    Set $\alpha_{l+1} = \tau \alpha_l$
    Set $l = l + 1$
until *terminal condition $f(x_k + \alpha_l p_k) \leq f(x_k) + \eta \alpha_l g_k^T p_k$ is satisfied;*
Output: approximate minimizer x_k

Lemma 6.2.8 (existence of step length satisfying Armijo condition will always satisfied in). *Suppose that*

- $f \in C^1$ *and* $g(x)$ *is Lipschitz continuous with Lipschitz constant* $\gamma(x)$
- p *is a descent direction at* x

Then for any given $\eta \in (0,1)$*, the Armijo condition*

$$f(x+\alpha p) \leq f(x) + \eta \alpha g(x)^T p$$

is satisfied for all $\alpha \in [0, \alpha_{max}]$*, where*

$$\alpha_{max} = \frac{2(\eta-1)g(x)^T p}{\gamma(x)\|p\|_2^2} > 0.$$

Proof.

$$\begin{aligned} f(x+\alpha p) &\leq f(x) + \alpha g(x)^T + \frac{1}{2}\gamma(x)\alpha^2\|p\|_2^2 \\ &\leq f(x) + \alpha g(x)^T p + \alpha(\eta-1)g(x)^T p \\ &= f(x) + \alpha\eta g(x)^T p \end{aligned}$$

where we use the fact that

$$\alpha \leq \frac{2(\eta-1)g(x)^T p}{\gamma(x)\|p\|_2^2}.$$

□

Remark 6.2.4 (backtracking algorithm will terminate in finite steps). Because the backtracking algorithm is always shrinking the step-length, it will eventually terminated with an α falling inside $[0, \alpha_{max}]$.

6.2.3.4 *Wolfe condition*

A more complex step size condition is the Wolfe condition, which requries both sufficient decrement of objective value and the slope.

Definition 6.2.4 (Wolfe conditions). *Given the current iterate* x_k*, search direction* p_k*, and constants* $0 < c_1 < c_2 < 1$*, we say that the step length* α_k *satisfies the* ***Wolfe conditions*** *if*

$$\begin{aligned} f\left(x_k + \alpha_k p_k\right) &\leq f\left(x_k\right) + c_1 \alpha_k \nabla f\left(x_k\right)^T p_k \\ \nabla f\left(x_k + \alpha_k p_k\right)^T p_k &\geq c_2 \nabla f\left(x_k\right)^T p_k \end{aligned}$$

Lemma 6.2.9 (Wolfe conditions for positive definite matrix in BFGS update). *Consider BFGS update where $B_k > 0$ and p_k is a descent direction for f at x_k. If α_k satisfies Wolfe condition such that $x_{k+1} = x_k + \alpha_k p_k$, then*

$$y_k^T s_k > 0,$$

where $s_k = (x_{k+1} - x_k)$ and $y_k = g_{k+1} - g_k$.

With such condition satisfied, $B_{k+1} > 0$.

Proof. From the Wolfe condition, we have

$$\nabla f\left(x_k + \alpha_k p_k\right)^T p_k \geq c_2 \nabla f\left(x_k\right)^T p_k.$$

Multiplying both sides by α_k and using $x_{k+1} = x_k + \alpha_k p_k$, we have

$$g_{k+1}^T s_k \geq c_2 g_k^T s_k,$$

and

$$g_{k+1}^T s_k - g_k^T s_k \geq c_2 g_k^T s_k - g_k^T s_k.$$

We then have

$$y_k^T s_k \geq \left(c_2 - 1\right) g_k^T s_k = \alpha_k \left(c_2 - 1\right) g_k^T p_k > 0$$

due to the fact hat $c_2 \in (0, 1)$ and $g_k^T p_k < 0$ (p_k is a descent direction). □

6.2.4 Complete algorithms

In the following, we give three complete algorithms that combine descent direction and step size computations. Note that Quasi-Newton method has to be used to Wolfe

condition to guarantee positive definiteness of B_{k+1}. A comprehensive convergence analysis can be found in [2].

Algorithm 3: Steepest decent Backtracking-Armijo line search algorithm

Input: Initial guess x_0

Set $k = 0$.

repeat

  Compute a steepest decent direction $p_k = -g_k$ at x_k.

  Compute a suitable step length α_k use Backtracking-Armijo line search algorithm

  Set $x_{k+1} = x_k + \alpha_k p_k$

  Set $k = k + 1$

until $\|\nabla f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$;

Output: approximate minimizer x_k

Algorithm 4: Modified Newton Backtracking-Armijo line search algorithm

Input: Initial guess x_0

Set $k = 0$.

repeat

  Set a decent direction $p_k = -B_k^{-1} g_k$ at x_k, where B_k is a symmetric positive definite matrix modified from Hessian H_k [Methodology 6.2.1].

  Compute a suitable step length α_k use Backtracking-Armijo line search algorithm

  Set $x_{k+1} = x_k + \alpha_k p_k$

  Set $k = k + 1$

until $\|\nabla f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$;

Output: approximate minimizer x_k

Algorithm 5: Quasi Newton with Wolfe line search algorithm

Input: Initial guess x_0, and inital positive definite B_0 approximate to the Hessian.

Set $k = 0$.

repeat

  Get a decent direction $p_k = -B_k^{-1} g_k$ at x_k.

  Compute a suitable step length α_k satisfies Wolfe condition.

  Set $x_{k+1} = x_k + \alpha_k p_k$

  Update

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k^T}{s_k^T B_k s_k}$$

  where $s_k = (x_{k+1} - x_k)$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$.

  Set $k = k + 1$

until $\|\nabla f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$;

Output: approximate minimizer x_k

6.3 Trust region method

6.3.1 Motivation and the framework

In each iteration, line search methods uses a proxy model to generate a descent direction, and determine a suitable step size that would decrease the objective function sufficiently along this direction. Trust region methods, in some ways, are setting a step size limit (determined by the so-called **trust region**) first, then find a minimizer from a proxy model that aims to decrease the value of the objective function. In Trust region methods, the proxy problem to be solve is defined as follows.

Definition 6.3.1 (trust-region subproblem). *The trust-region subproblem at kth iterate is*

$$\min_{x\in\mathbb{R}^n} m_k(s) = f_k + g_k^T s + \frac{1}{2}s^T B_k s, \text{ subject to } \|s\| \leq \delta_k$$

*where $f_k = f(x_k), g_k = \nabla f(x_k)$, B_k is a symmetric matrix and $\delta_k > 0$ is the **trust region radius**, and the norm is 2-norm.*

In trust region methods, in general we want B_k to approximate the Hessian $\nabla^2 f$ even the Hessian is not positive definite. In line search methods, we need B_k to be positive definite to generate descent directions.

The size of the trust region δ_k plays a critical role in the efficiency of algorithm since it bounds how far the iterate can move in each step. If chosen too small, the iterate cannot move too much each step although the proxy model well approximates the original objective function. If chosen too large, the proxy model might be a poor approximate to the original objective function, and a minimizer from a poor proxy model would not help find the true minimizer.

We measure the performance of current step by the reduction ratio of actual objective function reduction over the proxy model reduction.

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

The value of ρ_k provides important information that we could leverage to adaptively adjust the trust region radius. There are following situations.

- If ρ_k is negative or small then a threshold η_1, then $f(x_k + p_k) > f(x_k)$, which is undesirable, and we reject current step and re-search a new minimizer in a smaller range in next iteration.

- If ρ_k is greater than a threshold $\eta_2, \eta 1 < \eta_2 < 1$, then the proxy model excellently agrees with the true objective function, and we can expand the search range in the next iteration.
- If ρ_k is positive but fall below η_1, then the proxy model reasonably agrees with the true objective function, and we keep the current search range in the next iteration.

A generic trust-region algorithm is given by algorithm 6. An intuition on the convergence of the algorithm is: At sufficiently small δ, quadratic proxy model m is always a good proxy model to f. As the iterate continues to decrease the value of m, eventually a local minimum will be reached.

Algorithm 6: A generic trust-region algorithm

Input: Initial guess x_0

Choose $\delta_0 > 0, 0 < \gamma_d < 1 < \gamma_i$, and $0 < \eta_1 \leq \eta_2 < 1$

Set $k = 0$

repeat

  Compute (exactly or approximately) a search direction p_k as the solution to

$$\min_{x \in \mathbb{R}^n} m_k(s) = f_k + g_k^T s + \frac{1}{2} s^T B_k s, \text{ subject to } \|s\| \leq \delta_k.$$

  Set $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{\Delta m_k(s_k)}$

  if $\rho_k \geq \eta_2$ **then**

    set $x_{k+1} = x_k + s_k$ and $\delta_{k+1} = \gamma_i \delta_k$. (very successful)

  if $\rho_k > \eta_1$ **then**

    set $x_{k+1} = x_k + s_k$ and $\delta_{k+1} = \delta_k$ (successful)

  else

    set $x_{k+1} = x_k$ and $\delta_{k+1} = \gamma_d \delta_k$. (unsuccessful)

  end

  Set $k = k + 1$

until $\|\nabla f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$;

Output: approximate minimizer x_k

6.3.2 Cauchy point method

The Cauchy point method is to solve an minimizer of the optimization problem

$$\min_{x \in \mathbb{R}^n} m_k(s) = f_k + g_k^T s + \frac{1}{2} s^T B_k s, \text{ subject to } \|s\| \leq \delta_k$$

along the steepest descent direction $-g_k$.

More formally, we can define the following Cauchy point subproblem.

Definition 6.3.2 (Cauchy point of the trust-region subproblem). *The Cauchy point method is to solve the subproblem given by*

$$\min_{\alpha \geq 0} m_k(-\alpha g_k), \text{ subject to } \alpha \|g_k\| \leq \delta_k,$$

where $m_k(x) = f_k + g_k^T x + \frac{1}{2} x^T B_k x$.

The solution α_k^C *gives the* ***Cauchy point****:*

$$s_k^C = -\alpha_k^C g_k.$$

Lemma 6.3.1 (Cauchy point solution). *Let* $m_k(s)$ *be a trust-region subproblem and let* s_k^C *be the Cauchy point. Further denote* $\Delta m_k(s) = m_k(0) - m_k(s)$, *then*

$$\alpha_k = \begin{cases} \frac{\delta_k}{\|g_k\|} & \text{if } g_k^T B_k g_k \leq 0 \\ \min\left(\|g_k\|^2 / \left(g_k^T B_k g_k \right), \frac{\delta_k}{\|g_k\|} \right) & \text{otherwise} \end{cases}$$

Proof. Note that

$$m_k(-\alpha g_k) = f_k - \alpha \|g_k\|_2^2 + \frac{1}{2} \alpha^2 g_k^T B_k g_k.$$

If $g_k^T B_k g_k < 0$, m_k will continue to decrease as we increase α. Therefore the minimum will be found at the trust-region boundary; that is

$$a_k^C = \frac{\delta_k}{\|g_k\|}.$$

If $g_k^T B_k g_k < 0$, the minimum of m_k will be found either inside the trust region or at the trust-region boundary. Let's first assume the constraint does not exist, then the minimum is obtained by solving a one-dimensional quadratic optimization problem, which has solution given by

$$a_k^* = \frac{\|g_k\|^2}{g_k^T B_k g_k}.$$

If $a_k^* \|g_k\| \geq \delta_k$, the Cauchy point will lie on the boundary. We have

$$a_k^C = \frac{\delta_k}{\|g_k\|}.$$

If $a_k^* \|g_k\| < \delta_k$, the Cauchy point will lie within the boundary. We have

$$a_k^C = a_k^* = \frac{\|g_k\|^2}{g_k^T B_k g_k}.$$

□

As we can see, the Cauchy point barely utilize the curvature information on the matrix B_k in calculating the direction; in fact, it is used only to determine step length. Inherently, Cauchy point method is the steepest descent method under the umbrella of trust-region method and is expected to surfer from issues of the steepest descent method.

6.3.3 Exact solution method

In this section, we study the theory used to find out the exact solution of the trust-region subproblem. In practice, exact method is rarely used except for simple low-dimensional problems.

Theorem 6.3.1 (global minimizer condition for trust-region subproblem). *[2, p. 90] A vector s^* is a **global** minimizer of*

$$\min_{s \in \mathbb{R}^n} m(s) = f + s^T g + \frac{1}{2} s^T B s, \text{ subject to } \|s\|_2 \leq \delta$$

if and only if $\|s^*\| \leq \delta$ and there exists a scalar $\lambda^* \geq 0$ such that

- $(B + \lambda^* I)s^* = -g$
- *$B + \lambda^* I$ is positive semi-definite*
- *(complementary slackness)* $\lambda^*(\|s\|_2 - \delta) = 0$

Moreover, if $B + \lambda^* I$ is positive definite, then s^* is unique.

Proof. See the reference for the full proof. More details on a weaker condition can be found in KKT theory [Theorem 7.4.3 and Theorem 7.4.4]. □

6.3.4 Approximate method

Exact methods are usually quite expensive as it often involves another iterative routine to solve the subproblem. The problem is, even with the exact solution, the iterate could get rejected if the trust-region radius is too large. On the other hand, the Cauchy point method, which simply steepest descent, could be inherently slow.

In this section, we introduce approximate methods that lies in between the Cauchy point method and the exact solution.

The dogleg method finds an approximate solution by replacing the curved trajectory for $p(\delta)$ with a path consisting of two line segments

- The first line segment runs from the origin to the minimizer of m along the steepest descent direction, where
$$p^{\mathrm{U}} = -\frac{g^T g}{g^T B g} g.$$
- The second line segment runs from p^U to p^B, where
$$p^B = -B^{-1} g.$$

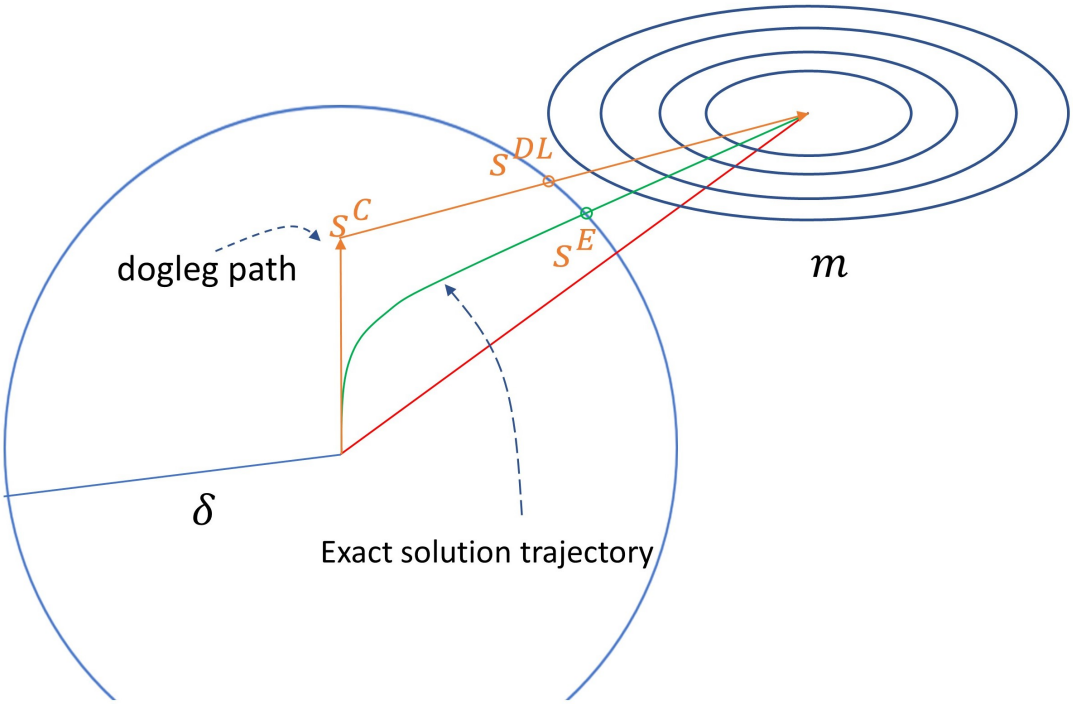


Figure 6.3.1: Demonstration of the dogleg path as an approximation to the exact solution path in the trust-region subproblem.

$$\tilde{p}(\tau) = \begin{cases} \tau p^{\mathrm{U}}, & 0 \leq \tau \leq 1 \\ p^{\mathrm{v}} + (\tau - 1)\left(p^{\mathrm{B}} - p^{\mathrm{U}}\right), & 1 \leq \tau \leq 2 \end{cases}$$

The minimizer alone this path can be found easily [2, p. 80].

If the trust region is large enough, then the minimizer will likely fall on the second segment that contains curvature information; otherwise, the minimizer will fall on the first segment, similar to Cauchy point method.

6.4 Conjugate gradient method

6.4.1 Motivating problems

Definition 6.4.1 (Problem of interest). *Given a symmetric positive-definite matrix A, solve the linear system*

$$Ax = b$$

which is equivalent to finding the unique minimizer of

$$\min_{x \in \mathbb{R}^n} q(x) = \frac{1}{2} x^T A x - b^T x,$$

which has necessary condition $\nabla q(x) = Ax - b = 0$ (due to strong convexity of q and Theorem 9.5.2].

On approach, known as coordinate descent, is to minimize each (orthogonal) dimension one at a time, and hopefully solve a n dimension problems in n steps [Figure 6.4.1(a)]. This is exactly the case when A is diagonal such that the columns of A are colinear with standard vectors, and we minimize along each column vector/coordinate every iteration. When A is not diagonal, coordinate descent can take iterations far than n to converge to the true solution.

On the other hand, we can perform eigendecomposition on A and then optimize along each eigenvector direction. However, this method is unlikely to scale to large systems due to the prohibitive cost of eigendecomposition.

Conjugate gradient method aims to seeks n such special direction to optimize in an efficient way.

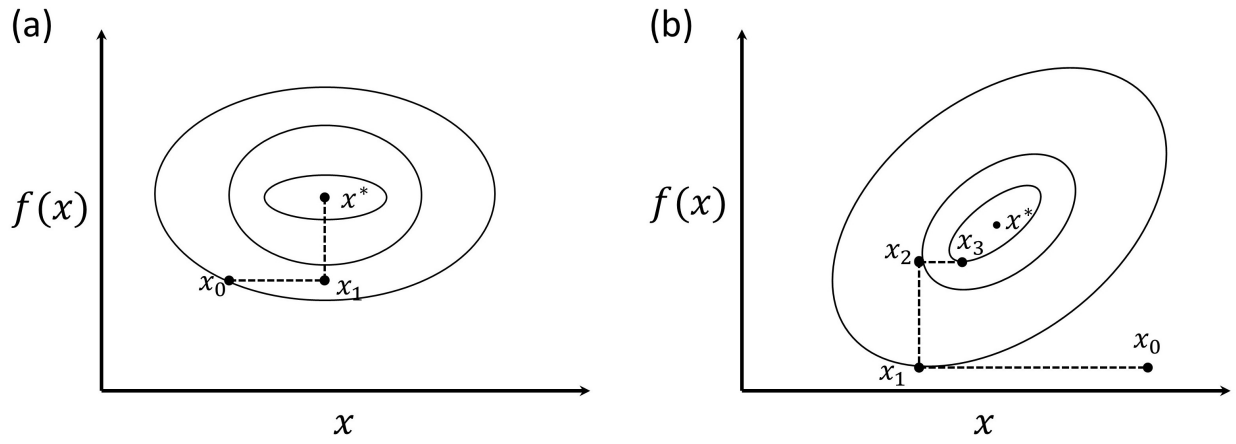


Figure 6.4.1: Demonstration of coordinate descent procedures when A is diagonal and non-diagonal.

6.4.2 Theory conjugate direction

Definition 6.4.2 (A conjugate directions). *A set of nonzero vector $\{s_0, s_1, ..., s_{n-1}\}$ is said to be conjugate with respect to the symmetric positive-definite matrix A if*

$$s_i^T A s_j = 0, \forall i \neq j$$

Remark 6.4.1 (eigenvectors are A conjugate directions, but not reverse).

- Let A be symmetric positive-definite matrix, then we know that eigenvectors corresponding to distinct eigenvalues are orthogonal to each other.

$$v_i A v_j = v_i \lambda_j v_j = 0$$

- For a set of vectors $v_1, v_2, ..., v_n$ are A conjugate directions, then these vectors are eigenvectors.

Lemma 6.4.1 (conjugate directions are linearly independent). *Any set of vectors $\{p_1, ..., p_n\}$ that are A (A is symmetric positive-definite) conjugate directions will be linearly independent set.*

Proof. Suppose they are linear dependent. WLOG, we have $p_1 = \sum_{j=2}^{n} a_j p_j$. Multiply both sides by $(Ap_1)^T$, we get $p_1^T A p_1 > 0$ on the left side. Then the right hand side equals o due to the conjugation. □

Lemma 6.4.2 (expanding subspace minimization). *[2, p. 103][1, p. 121] For any $x_0 \in \mathbb{R}^n$ the sequence $\{x_k\}$ generated via conjugate gradient method are expanding subspace minimizers. That is*

$$x_{k+1} = \arg\min_{x \in \mathcal{M}_k} f(x),$$

where

$$\mathcal{M}_k = \{x | x = x_0 + v, v \in span\{p_0, p_1, ..., p_k\}.$$

Moreover, the iteration x_k converges to the unique minimizer x^* at most n steps.

Proof.

$$\frac{\partial f(x_i + \alpha p_i)}{\partial \alpha} = \nabla f(x_{i+1})^T p_i = 0.$$

For $i = 0, ..., k-1$, we have

$$\begin{aligned}
\nabla f(x_{k+1})^T p_i &= (Ax_{k+1} - b)^T p_i \\
&= (x_i + \sum_{j=i+1}^{k} \alpha_j p_j)^T A p_i - b^T p_i \\
&= x_i^T A p_i - b^T p_i = \nabla f(x_i)^T p_i = 0.
\end{aligned}$$

Therefore,

$$\frac{f(x_{k+1} + \beta_1 p_1 + ... + \beta_k p_k)}{\partial \beta_i}|_{\beta_1 = \beta_2 = ... = 0} = \nabla f(x_{k+1})^T p_i = 0.$$

□

6.4.3 Linear conjugate gradient algorithm

Definition 6.4.3 (conjugate direction generation algorithms). *Given a starting point $x_0 \in \mathbb{R}^n$ and a set of conjugate directions $\{p_0, p_1, ..., p_{n-1}\}$, let us generate the sequence $\{x_k\}$ by setting*

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

where α_k is the one-dimensional minimizer of the quadratic function ϕ along $x_k + \alpha p_k$, given explicitly by

$$\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}, r_k = Ax_k - b.$$

Lemma 6.4.3. *[2, p. 103] For any $x_0 \in \mathbb{R}^n$ the sequence $\{x_k\}$ generated via conjugate direction algorithms converges to the solution x^* of the linear system $Ax = b$ in at most n steps.*

Proof. Since the directions $\{p_i\}$ are linearly independent, they must span the whole space $\mathbb{R}^n$. □

Remark 6.4.2 (special cases: conjugate directions are eigenvectors)**.** If we use eigenvector as conjugate directions, then we can see that we are essentially doing sequential optimization on a set of orthonormal basis. The condition of finality [Theorem 5.4.3] guarantees we arrive at the optimum at n steps.

Lemma 6.4.4 (expanding subspace minimization). *[2, p. 103] For any $x_0 \in \mathbb{R}^n$ the sequence $\{x_k\}$ generated via conjugate direction algorithms converges to the solution x^* of the linear system $Ax = b$ in at most n steps.*

Algorithm 7: A linear conjugate algorithm

Input: Initial guess x_0

Set $k = 0$, $r_0 = Ax_0 - b$, $p_0 = -r_0$.

repeat

Compute step size

$$\alpha_k = \frac{r_k^T r_k}{p_k^T A p_k}$$

Update residual via

$$x_{k+1} = x_k + \alpha_k p_k$$
$$r_{k+1} = r_k + \alpha_k p_k$$

Generate new conjugate direction

$$\beta_{k+1} = r_{k+1}^T r_{k+1}$$
$$p_{k+1} = -r_{k+1} + \beta_{k+1} p_k$$

Set $k = k + 1$

until r_k *sufficiently small*;

Output: approximate minimizer x_k

6.5 Least square problems

6.5.1 Linear least square theory and algorithm

6.5.1.1 *Linear least square problems*

Definition 6.5.1 (linear least square problem). *Given a $A \in \mathbb{R}^{m\times n}, b \in R^m$, find a x that solves*

$$\min_x f(x) = \frac{1}{2}\|Ax - b\|^2$$

Remark 6.5.1.

1. We usually assume $m \geq n$, then this is a over-determined system(can be consistent or inconsistent); If $m < n$, there will be infinitely solutions(assuming consistence).
2. $f(x) = \frac{1}{2}b^Tb - x^TA^Tb + \frac{1}{2}x^TA^TAx$
3. $f(x)$ is convex, since for any matrix A, we always have $A^TA \geq 0$.
4. If A has full column rank, then $A^TA > 0$, indicating f is strictly convex. Note that for any matrix A, we always have $A^TA \geq 0$, when A has full column rank, then Ax is o only when x is o, there fore $(Ax)^T(Ax) > 0, \forall x \neq 0$.
5. $\nabla f = 0 \Rightarrow A^TAx - A^Tb = 0$

Remark 6.5.2 (extreme value property, existence and uniqueness, global vs. local).

- (existence)The minimal value always exists, and $f_{min} \geq 0$.
  - If $Ax = b$ is consistent, then $f_{min} = 0$.
  - If $Ax = b$ is inconsistent, then f_{min} will be the minimum distance of vector b to the subspace spanned by columns of A.
- (uniqueness) uniqueness of the minimizer depends on the rank of A (no matter consistence).
  - If $m < n$, there will be infinitely many minimizers.
  - If $m \geq n$, there will be infinitely many minimizers(when A has non-trivial null spaces) or unique minimizer(when A has full column rank).
- Any local minimizer x^* such that $\nabla f(x^*)$ will be a global minimizer due to convexity.

Remark 6.5.3 (solution methods).

- Direct methods involve directly solving $A^TAx = A^Tb$ using LU, Cholesky, QR, SVD, and linear CG methods under the assumption of $A^TA > 0$. For details, see [3].
- For recursive methods, see subsection 17.5.3.

6.5.1.2 *SVD methods*

Lemma 6.5.1 (minimum error solution via SVD theory, recap). *Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Let A have SVD decomposition Theorem 4.9.1 given by*

$$A = [U_1\ U_2] \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} V_1^T \\ V_2^T \end{bmatrix}.$$

Then the minimizers of

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2$$

is a set

$$x^* = V_1\Sigma^{-1}U_1^T b + V_2 y.$$

Among the set, the element with the minimum 2-norm length is

$$x_m^* = V_1\Sigma^{-1}U_1^T b.$$

Proof. Lemma 4.1.9. □

Note 6.5.1 (steps). The above method can be executed using the following procedures:

- Compute economic SVD

$$A = U\Sigma V^T = \sum_{i=1}^{n} \sigma_i u_i v_i^T$$

- Form $y = U^T b$
- Form $z = \Sigma^{-1} y$
- $x^* = Vz$. Note that x^* is unique.

Remark 6.5.4. See more discussions from the perspective of linear equations, see subsection 4.1.5.

6.5.1.3 *Extension to L^p norm optimization*

Definition 6.5.2 (L^p norm linear least square problem). *Given a $A \in \mathbb{R}^{m\times n}, b \in R^m$, find a x that solves*

$$\min_x f(x) = \frac{1}{2}\|Ax - b\|_p,$$

where $\|\cdot\|_p$ is the $p-$norm for a vector.

Algorithm 8: Iteratively reweighted least squares for p norm least square

Input: A small threshold number ϵ, a large value w_{big} for weighting zero residuals

1 Set $k = 0$

2 Compute the diagonal matrix W with diagonal element being

$$W_{ii} = p(x_i;\beta)(1 - p(x_i;\beta)), i = 1, 2, ..., N.$$

3 **repeat**

4 compute the diagnoal matrix

$$W^k : w_i^k = \left|y_i - x_i^T b^{(k)}\right|^{2-p},$$

if $\left|y_i - x_i^T b^{(k)}\right| < \epsilon$, set $w_i^{(k)} = w_{big}$.

5 Compute $b^{(k+1)}$ by solving the weighted least squares problem: minimize $(y - Xb)^T W^{(k)}(y - Xb)$.

6 set $k = k + 1$.

7 **until** *stopping criteria* $\left\|b^{(k+1)} - b^{(k)}\right\| \leq \epsilon$ *is met*;

Output: the optimalized value b.

Remark 6.5.5 (interpretation the weight calculation).

- The algorithm is in [4, p. 233].
- We can formulate the pth power of the norm as

$$\|y - Xb\|_p^p = (y - Xb)^T W(y - Xb),$$

where $W = diag(\left|y_1 - x_1^T b\right|^{2-p}, \left|y_2 - x_2^T b\right|^{2-p}, ..., \left|y_n - x_n^T b\right|^{2-p})$.

6.5.2 nonlinear least square problem

Definition 6.5.3 (Nonlinear least square problem). *Given a function $F : \mathbb{R}^n \to \mathbb{R}^m$, the nonlinear least square problem is to find a vector that solves the optimization problem*

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2}\|F(x)\|^2$$

Remark 6.5.6. **Notes:**

- we will assume $m \geq n$
- $\nabla f = J^T(x)F(x)$, where

$$J(x) = \nabla F(x) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_1}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \frac{\partial F_m}{\partial x_2} & \cdots & \frac{\partial F_m}{\partial x_n} \end{pmatrix}$$

- $\nabla^2 f = J^T J + \sum_{i=1}^m \nabla^2 F_i(x) F_i(x)$
- f is typically nonconvex
- x is the first-order solution if it satisfies

$$\nabla f = J^T(x)F(x) = 0$$

6.5.3 Line search Gauss-Newton method

Definition 6.5.4 (Gauss-Newton subproblem). *The Gauss-Newton subproblem to compute p as a minimizer of*

$$\min_{p \in \mathbb{R}^n} \frac{1}{2}\|F(x) + J(x)p\|_2^2 = \frac{1}{2}\|F(x)\|_2^2 + p^T J(x)^T F(x) + \frac{1}{2}p^T J(x)^T J(x) p,$$

where x is given.

Lemma 6.5.2 (properties of Gauss-Newton subproblem). *For a Gauss-Newton subproblem, if J is full column rank, then*

- *$J^T J$ is positive definite and the Gauss-Newton problem has a unique minimizer.*
- *If $\nabla f(x) = J(x)^T F(x) \neq 0$, then the minimizer p_G satisfy*

$$p_G^T \nabla f(x) < 0;$$

that is, p_G is a descent direction to the original optimization problem.

Proof. (1) When J^TJ is positive definite, the Gauss-Newton problem is a convex optimization problem. (2) $p_G\nabla f(x) = p_G J(x)^T F(x) = -p_G J(x)^T J(x) p_G < 0$, where we use $F(x) = Jp_G$. □

Algorithm 9: Gauss-Newton method for nonlinear least-square algorithm

Input: Initial guess x_0

Set $k = 0$ **repeat**

  Compute a search direction p_k as the solution to

$$\min_{p\in\mathbb{R}^n} \frac{1}{2}\|F(x_k) - J(x_k)p\|_2^2$$

  Compute a suitable step length α_k use Backtracking-Armijo line search algorithm

  Set $x_{k+1} = x_k + \alpha_k p_k$

  Set $k = k + 1$

until $\|\nabla f(x_k)\| \leq 10^{-8}\max(1, \|\nabla f(x_0)\|)$;

Output: approximate minimizer x_k

Remark 6.5.7 (interpretation)**.**

- We require $J(x_k)$ to be full column rank in the whole process such that p_G can be guaranteed to be descent direction.
- We can also use the steepest descent direction $p = -\nabla f(x) = -J(x)^T F(x)$ as the descent direction; however, the Gauss Newton direction p_G is better because it contains curvature information.

6.5.4 Trust region method

Algorithm 10: Levenberg-Marquardt method for nonlinear least-square algorithm

Input: Initial guess x_0

Choose $\delta_0 > 0, 0 < \gamma_d < 1 < \gamma_i$, and $0 < \eta_s \leq \eta_{vs} < 1$ Set $k = 0$ **repeat**

  Compute a search direction p_k as the solution to

$$\min_{p \in \mathbb{R}^n} \frac{1}{2} \|F(x_k) - J(x_k)p\|_2^2$$

  Set $\rho_k = \frac{f(x_k) - f(x_k + s_k)}{\Delta m_k(s_k)}$

  if $\rho_k \geq \eta_{vs}$ **then**

    set $x_{k+1} = x_k + s_k$ and $\delta_{k+1} = \gamma_i \delta_k$

  if $\rho_k > \eta_s$ **then**

    set $x_{k+1} = x_k + s_k$ and $\delta_{k+1} = \delta_k$

  else

    set $x_{k+1} = x_k + s_k$ and $\delta_{k+1} = \gamma_d \delta_k$

  end

  Set $k = k + 1$

until $\|\nabla f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$;

Output: approximate minimizer x_k

Remark 6.5.8 (interpretation).

- The trust region subproblem here is slightly different from general trust region subproblem which directly use Hessian as the quadratic term.
- The trust region Levenberg-Marquardt algorithm is better than the linear search Gauss-Newton algorithm in handling cases where $J(x_k)$ can have dependent columns.

6.5.5 Application: roots for nonlinear equation

Definition 6.5.5 (roots for nonlinear equation problem). *Given a function $\boldsymbol{F} : \mathbb{R}^n \to \mathbb{R}^n$, and assume $\boldsymbol{F}$ is at least continuously differentiable, find a vector $x^* \in \mathbb{R}^n$ such that*

$$\boldsymbol{F}(x^*) = 0$$

Such x^* is called a root.

If x and F are both one-dimensional, we can use following Newton method.

Algorithm 11: Newton method for root finding

Input: Initial guess x_0

Set $k = 0$ **repeat**

  $x_{k+1} = x_k - \frac{f(x_k)}{f'_k}$

  Set $k = k + 1$

until $\|f(x_k)\| \leq 10^{-8} \max(1, \|\nabla f(x_0)\|)$;

Output: approximate root x_k

Remark 6.5.9 (Interpretation of classic Newton's method).

- The new iterate x_{k+1} is generated by setting the first moder $f(x_k + s) = f(x_k) + f'(x_k)s = 0, s = x_{k+1} - x_k$.
- If $\boldsymbol{F}$ is continuously differentiable, then use Newton's method can converge to the root if initial x_0 starts close enough.

Lemma 6.5.3. *Consider the optimization problem*

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} \|F(x)\|^2,$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ *then*

- *Let x be the root of $F(x)$, then x is the minimizer of the optimization problem.*
- *If x satisfies $\nabla f(x) = 0$, and x^** ***is not degenerate***[a]*, then x^* is the root of $F(x)$.*

[a] $x^* \in \mathbb{R}^n$ is called degenerate if the Jacobian $J(x^*)$ is singular.

Proof. (1) If $F(x^*) = 0$, then x^* is a minimizer since $f(x) \geq 0, \forall x$; (2) At local minimizer, we must have $\nabla f = J(x)^T F(x) = 0$, if $J(x^*) \neq 0$, then we must have $F(x) = 0$. □

Remark 6.5.10 (calculating root by minimizing nonlinear least square).

- This lemma enables us to convert nonlinear root problem to nonlinear least square problem.
- After converting to nonlinear least problem, we have two corresponding algorithms (Gauss-Newton algorithm and Levenberg-Marquardt algorithm) to solve the problem.

6.6 Notes on bibliography

Good general references are [1][2][3].

A good reference on Least square problems are [5].

BIBLIOGRAPHY

1. Bertsekas, D. *Nonlinear programming* ISBN: 9781886529007 (Athena Scientific, 2016).
2. Nocedal, J. & Wright, S. *Numerical optimization* (Springer Science & Business Media, 2006).
3. Robinson, D. *Nonlinear optimization I lecture notes* (Johns Hopkins University, 2015).
4. Gentle, J. *Matrix Algebra: Theory, Computations and Applications in Statistics* ISBN: 9783319648675 (Springer International Publishing, 2017).
5. Pighin, F. & Lewis, J. P. *Practical least-squares for computer graphics: Video files associated with this course are available from the citation page* in *ACM SIGGRAPH 2007 courses* (2007), 1–57.