Lecture 7
Least Squares

Lin ZHANG, PhD
School of Software Engineering
Tongji University
Fall 2020
Outline

• Why is least squares an important problem?
• Linear Least Squares
• Non-linear Least Squares
Why is least squares an important problem?

In intelligent automobile industry, some mathematical terminologies are often met:

- Homogeneous linear equation system
- Inhomogeneous linear equation system
- Jacobian matrix
- Hessian matrix
- Lagrange multiplier
- Line search
- Steepest descent method
- Newton method
- Damped method
- Trust-region method
- Damped Newton method
- Gauss-Newton method
- Levenberg-Marquardt method
- Dog-leg method
Why is least squares an important problem?

Ex1: bird’s-eye-view calibration

bird’s-eye-view image  original perspective image
Why is least squares an important problem?

We need to estimate the homography between the image plane and the physical plane. This is achieved by an offline calibration process.

\[ \{x_i \leftrightarrow u_i\}_{i=1}^n \]

A point on the physical plane \quad A point on the image plane

We know there existing an \( H \) satisfying

\[ x_i = Hu_i \]

We need to find \( H \) from \( \{x_i \leftrightarrow u_i\}_{i=1}^n \)
Why is least squares an important problem?

For one point pair $x \leftrightarrow u$, we have

$$
egin{pmatrix}
  x \\
  y \\
  1
\end{pmatrix}
=

\begin{bmatrix}
  h_{11} & h_{12} & h_{13} \\
  h_{21} & h_{22} & h_{23} \\
  h_{31} & h_{32} & h_{33}
\end{bmatrix}

\begin{pmatrix}
  u \\
  v \\
  1
\end{pmatrix}

\Rightarrow

\begin{align*}
  h_{11}u + h_{12}v + h_{13} &= sx \\
  h_{21}u + h_{22}v + h_{23} &= sy \\
  h_{31}u + h_{32}v + h_{33} &= s
\end{align*}

\Rightarrow

\begin{pmatrix}
  h_{11} \\
  h_{12} \\
  h_{13} \\
  h_{21} \\
  h_{22} \\
  h_{23} \\
  h_{31} \\
  h_{32} \\
  h_{33}
\end{pmatrix}

\begin{pmatrix}
  u \\
  v \\
  1
\end{pmatrix}

= 0

\Rightarrow

Since we have $n$ point pairs, we get

$$
A_{2n \times 9} h_{9 \times 1} = 0
$$

How to solve this **homogeneous** linear equation system?
Why is least squares an important problem?

Since only the ratios among the elements of $H$ take effect, in another way we can fix $h_{33}=1$,

$$
\begin{pmatrix}
  x \\
y \\
1
\end{pmatrix}
= \begin{bmatrix}
h_{11} & h_{12} & h_{13} \\
h_{21} & h_{22} & h_{23} \\
h_{31} & h_{32} & 1
\end{bmatrix}
\begin{pmatrix}
u \\
v \\
1
\end{pmatrix}
\Rightarrow
\begin{cases}
h_{11}u + h_{12}v + h_{13} = x \\
h_{21}u + h_{22}v + h_{23} = y \\
h_{31}u + h_{32}v + 1 = 1
\end{cases}
\Rightarrow
\begin{cases}
h_{11}u + h_{12}v + h_{13} = x \\
h_{31}u + h_{32}v = x
\end{cases}
$$

Since we have $n$ point pairs, we get

$$
A_{2n \times 8} h_{8 \times 1} = b_{2n \times 1}
$$

How to solve this *inhomogeneous* linear equation system?
Why is least squares an important problem?

**Ex2: Camera calibration**

The widely used Zhang Zhengyou’s method actually needs to solve a non-linear minimization problem,

\[ A^*, R^*_i, t^*_i = \arg \min_{A, R_i, t_i} \sum_{i=1}^{n} \sum_{j=1}^{m} \| m_{ij} - \hat{m}(A, R_i, t_i, M_j) \|^2 \]

How to solve this non-linear minimization problem?
Why is least squares an important problem?

**Ex3: visual SLAM**

The core problem of visual slam is how to recover the poses of the camera from its observations (images).

One typical problem to solve in visual slam,

\[ \xi^* = \arg \min_{\xi} \sum_{i=1}^{n} \left\| \mathbf{u}_i - \frac{1}{s_i} \mathbf{K} \exp(\xi^\top) \mathbf{p}_i \right\|^2 \]

where \( \mathbf{p}_i \) is a feature point, \( \mathbf{u}_i \) is \( \mathbf{p}_i \)'s projection on the current frame; we need to identify the optimal pose \( \xi^* \) that best conforms to the observation.

How to solve this non-linear minimization problem?
Why is least squares an important problem?

- All these problems can be summarized as three kinds of problems

**Inhomogeneous linear equation system**

\[ \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{b} \neq \mathbf{0}, \mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{x} \in \mathbb{R}^{n \times 1}, \text{rank}(\mathbf{A}) = n, \text{rank}([\mathbf{A} \ \mathbf{b}]) = n + 1 \]

**Homogeneous linear equation system**

\[ \mathbf{A}\mathbf{x} = \mathbf{0}, \mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{x} \in \mathbb{R}^{n \times 1}, \text{rank}(\mathbf{A}) = n \]

**Non-linear least squares problem**

\[ \mathbf{x}^* = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{f}(\mathbf{x})\|^2 \]

where \( f_i(\mathbf{x}) \) is a nonlinear function of \( \mathbf{x} \).
Outline

• Why is least squares an important problem in autonomous driving?
• Linear Least Squares
  • LS for inhomogeneous linear system
  • LS for homogeneous linear system
• Non-linear Least Squares
Consider the following linear equations system

\[
\begin{align*}
    x_1 + x_2 &= 3 \\
    2x_1 + x_2 &= 4
\end{align*}
\]

Matrix form:

\[
A \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}
\]

It can be easily solved

\[
\begin{align*}
    x_1 &= 1 \\
    x_2 &= 2
\end{align*}
\]
LS for Inhomogeneous Linear System

How about the following one?

\[
\begin{cases}
    x_1 + x_2 = 3 \\
    2x_1 + x_2 = 4 \\
    x_1 + 2x_2 = 6
\end{cases} \iff 
\begin{bmatrix}
    1 & 1 \\
    2 & 1 \\
    1 & 2
\end{bmatrix} 
\begin{bmatrix}
    x_1 \\
    x_2
\end{bmatrix} = 
\begin{bmatrix}
    3 \\
    4 \\
    6
\end{bmatrix}
\]

It does not have a solution!

What is the condition for a linear equation system \( Ax = b \) can be solved?

*Can we solve it in an approximate way?*

*A: we can use least squares technique!*

Carl Friedrich Gauss
Let's consider a system of $p$ linear equations with $q$ unknowns

\[
\begin{align*}
    a_{11}x_1 + a_{12}x_2 + \ldots + a_{1q}x_q &= b_1 \\
    a_{21}x_1 + a_{22}x_2 + \ldots + a_{2q}x_q &= b_2 \\
    \vdots & \quad \vdots \\
    a_{p1}x_1 + a_{p2}x_2 + \ldots + a_{pq}x_q &= b_p
\end{align*}
\]

\[\iff \quad Ax = b\]

We consider the case: $p > q$, and $\text{rank}(A) = q$

In general case, there is no solution!

Instead, we want to find a vector $x$ that minimizes the error:

\[
E(x) \equiv \sum_{i=1}^{p} (a_{i1}x_1 + \ldots + a_{iq}x_q - b_i)^2 = \|Ax - b\|^2
\]
LS for Inhomogeneous Linear System

\[ x^* = \arg \min_{x} E(x) = \arg \min_{x} \|Ax - b\|_2^2 \]

\[ x^* = (A^T A)^{-1} A^T b \]

Pseudoinverse of A

How about the pseudoinverse of A when A is square and non-singular?
Let’s consider a system of $p$ linear equations with $q$ unknowns

\[
\begin{cases}
  a_{11}x_1 + a_{12}x_2 + \ldots + a_{1q}x_q = 0 \\
  a_{21}x_1 + a_{22}x_2 + \ldots + a_{2q}x_q = 0 \\
  \vdots \\
  a_{p1}x_1 + a_{p2}x_2 + \ldots + a_{pq}x_q = 0
\end{cases}
\]

$\iff A\mathbf{x} = \mathbf{0}$

We consider the case: $p > q$, and $\text{rank}(A) = q$

Theoretically, there is only a trivial solution: $\mathbf{x} = 0$

So, we add a constraint $\|\mathbf{x}\|_2 = 1$ to avoid the trivial solution
LS for Homogeneous Linear System

We want to minimize $E(x) = \|Ax\|_2^2$, subject to $\|x\|_2 = 1$

$$x^* = \arg \min_x E(x), \text{ s.t., } \|x\|_2 = 1$$

Use the Lagrange multiplier to solve it,

$$x^* = \arg \min_x \left[ \|Ax\|_2^2 + \lambda \left( 1 - \|x\|_2^2 \right) \right]$$

Solving the stationary point of the Lagrange function,

$$\begin{cases}
\frac{\partial}{\partial x} \left[ \|Ax\|_2^2 + \lambda \left( 1 - \|x\|_2^2 \right) \right] = 0 \\
\frac{\partial}{\partial \lambda} \left[ \|Ax\|_2^2 + \lambda \left( 1 - \|x\|_2^2 \right) \right] = 0
\end{cases}$$
Then, we have

$$\lambda \left( \frac{\partial}{\partial x} \left[ \| A x \|_2^2 + \lambda \left( 1 - \| x \|_2^2 \right) \right] \right) = 0$$

Then, we have

$$A^T A x = \lambda x$$

$x$ is the eigen-vector of $A^T A$ associated with the eigenvalue $\lambda$

$$E(x) = \| A x \|_2^2 = x^T A^T A x = x^T \lambda x = \lambda$$

The unit vector $x$ is the eigenvector associated with the minimum eigenvalue of $A^T A$
Outline

• Why is least squares an important problem in autonomous driving?
• Linear Least Squares
• Non-linear Least Squares
  • General Methods for Non-linear Optimization
    • Basic Concepts
    • Descent Methods
  • Non-linear Least Squares Problems
**Definition 1**: Local minimizer

Given $F : \mathbb{R}^n \mapsto \mathbb{R}$. Find $x^*$ so that

$$F(x^*) \leq F(x), \text{ for } \|x - x^*\| < \delta$$

where $\delta$ is a small positive number.
Basic Concepts

Assume that the function $F$ is differentiable and so smooth that the Taylor expansion is valid,

$$F(x + h) = F(x) + h^T F'(x) + \frac{1}{2} h^T F''(x) h + O(\|h\|^3)$$

where $F'(x)$ is the gradient and $F''(x)$ is the Hessian,

$$F'(x) = \begin{bmatrix} \frac{\partial F}{\partial x_1}(x) \\ \frac{\partial F}{\partial x_2}(x) \\ \vdots \\ \frac{\partial F}{\partial x_n}(x) \end{bmatrix}, \quad F''(x) = \begin{bmatrix} \frac{\partial^2 F}{\partial x_i \partial x_j}(x) \end{bmatrix}_{n \times n} = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1 \partial x_1} & \frac{\partial^2 F}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 F}{\partial x_n \partial x_n} \end{bmatrix}$$
Basic Concepts

Assume that the function $F$ is differentiable and so smooth that the Taylor expansion is valid,

$$F(x + h) = F(x) + h^T F'(x) + \frac{1}{2} h^T F''(x) h + O\left(\|h\|^3\right)$$

where $F'(x)$ is the gradient and $F''(x)$ is the Hessian,

It is easy to verify that,

$$F''(x) = \frac{dF'(x)}{dx^T}$$
**Theorem 1**: Necessary condition for a local minimizer

If $x^*$ is a local minimizer, then

$$F'(x^*) = 0$$

**Definition 2**: Stationary point

If $F'(x_s) = 0$,

then $x_s$ is said to be a stationary point for $F$.

A local minimizer (or maximizer) is also a stationary point. A stationary point which is neither a local maximizer nor a local minimizer is called a **saddle point**
Theorem 2: Sufficient condition for a local minimizer

Assume that $x_s$ is a stationary point and that $F''(x_s)$ is positive definite, then $x_s$ is a local minimizer.

If $F''(x_s)$ is negative definite, then $x_s$ is a local maximizer. If $F''(x_s)$ is indefinite (i.e., it has both positive and negative eigenvalues), then $x_s$ is a saddle point.
Outline

• Why is least squares an important problem in autonomous driving?
• Linear Least Squares

• Non-linear Least Squares
  • General Methods for Non-linear Optimization
    • Basic Concepts
    • Descent Methods
  • Non-linear Least Squares Problems
Descent Methods

- All methods for non-linear optimization are iterative: from a staring point $x_0$ the method produces a series of vectors $x_1, x_2, ..., x_k$, which (hopefully) converges to $x^*$

- The methods have measures to enforce the descending condition,

$$ F(x_{k+1}) < F(x_k) $$

Thus, these kinds of methods are referred to as “descent methods”

- For descent methods, in each iteration, we need to
  - Figure out a suitable **descent direction** to update the parameter
  - Find a **step length** giving good decrease in the $F$ value
Consider the variation of the $F$-value along the half line staring at $x$ and with direction $h$,

$$
F(x + \alpha h) = F(x) + \alpha h^T F'(x) + O(\alpha^2)
$$

$$
= F(x) + \alpha h^T F'(x) \quad \text{for sufficiently small } \alpha > 0
$$

**Definition 3:** Descent direction

$h$ is a descent direction for $F$ at $x$ if

$$
h^T F'(x) < 0
$$
Descent Methods

2-phase methods
(direction and step length are determined in 2 phases separately)

Phase I
Methods for computing descent direction
✓ Steepest descent method
✓ Newton’s method
✓ SD and Newton hybrid

Phase II
Methods for computing the step length
✓ Line search

1-phase methods
(direction and step length are determined jointly)
✓ Trust region methods
✓ Damped methods
  • Ex: Damped Newton method
2-phase methods: General Algorithm Framework

**Algo#1**: 2-phase Descent Method (a general framework)

```
begin
    k := 0; x := x_0; found := false          \{Starting point\}
    while (not found) and (k < k_{max})
        h_d := search_direction(x)          \{From x and downhill\}
        if (no such h exists)
            found := true                     \{x is stationary\}
        else
            a := step_length(x, h_d)          \{from x in direction h_d\}
            x := x + a h_d; k := k + 1      \{next iterate\}
    end
```
2-phase methods: steepest descent to compute the descent direction

When we perform a step $\alpha \mathbf{h}$ with positive $\alpha$, the relative gain in function value satisfies,

$$\lim_{\alpha \to 0} \frac{F(x) - F(x + \alpha \mathbf{h})}{\alpha \| \mathbf{h} \|} = \lim_{\alpha \to 0} \frac{F(x) - \left[ F(x) + \alpha \mathbf{h}^T \mathbf{F}'(x) \right]}{\alpha \| \mathbf{h} \|} = -\frac{\mathbf{h}^T \mathbf{F}'(x)}{\| \mathbf{h} \|}$$

$$= -\frac{\| \mathbf{h} \| \| \mathbf{F}'(x) \| \cos \theta}{\| \mathbf{h} \|} = -\| \mathbf{F}'(x) \| \cos \theta$$

where $\theta$ is the angle between vectors $\mathbf{h}$ and $\mathbf{F}'(x)$

This shows that we get the greatest relative gain when $\theta = \pi$, i.e., we use the steepest descent direction $\mathbf{h}_{sd}$ given by $\mathbf{h}_{sd} = -\mathbf{F}'(x)$

This is called the steepest gradient descent method
2-phase methods: steepest descent to compute the descent direction

- Properties of the steepest descent methods
  - The choice of descent direction is “the best” (locally) and we could combine it with an exact line search
  - A method like this converges, but the final convergence is linear and often very slow
  - For many problems, however, the method has quite good performance in the initial stage of the iterative; Considerations like this has lead to the so-called hybrid methods, which – as the name suggests – are based on two different methods. One of which is good in the initial stage, like the gradient method, and another method which is good in the final stage, like **Newton’s method**
2-phase methods: Newton’s method to compute the descent direction

Newton’s method is derived from the condition that $x^*$ is a stationary point, i.e.,

$$F'(x^*) = 0$$

From the current point $x$, along which direction moves how far (a vector $h_n$), will it be most possible to arrive at a stationary point? I.e., we solve $h_n$ from,

$$F'(x + h_n) = 0$$

what is the solution to $h_n$?
2-phase methods: Newton’s method to compute the descent direction

\[ F'(x + h) = \left[ \begin{array}{c} \frac{\partial F}{\partial x_1} \big|_{x+h} \\ \frac{\partial F}{\partial x_2} \big|_{x+h} \\ \vdots \\ \frac{\partial F}{\partial x_n} \big|_{x+h} \end{array} \right] + \left[ \begin{array}{c} \frac{\partial^2 F}{\partial x_1 \partial x_1} \big|_x \\ \frac{\partial^2 F}{\partial x_1 \partial x_2} \big|_x \\ \vdots \\ \frac{\partial^2 F}{\partial x_1 \partial x_n} \big|_x \end{array} \right] h = F'(x) + F''(x) h \]

So \( h_n \) is the solution to, \( F''(x) h_n = -F'(x) \)

Suppose that \( F''(x) \) is positive definite, then,
\[ h_n^T F''(x) h_n = -h_n^T F'(x) > 0 \]
i.e., \[ h_n^T F'(x) < 0 \]
indicates that \( h_n \) is a descent direction

In classical Newton method, the update is (then it can be regarded as a 1-phase method)
\[ x := x + h_n \]
However, in most modern implementations,
\[ x := x + \alpha h_n \]
where \( \alpha \) is determined by line search
2-phase methods: Newton’s method to compute the descent direction

• Properties of the Newton’s method
  – Newton’s method is very good in the final stage of the iteration, where \( x \) is close to \( x^* \)
  – Only when \( F'(x) \) is positive definite, it is sure that \( h_n \) is a descent direction
  – So, we can build a hybrid method, based on Newton’s method and the steepest descent method,

In Algo#1, we can use a hybrid method to get the descent direction

\[
\begin{align*}
\text{if } F''(x) \text{ is positive definite} \\
& h_d := h_n \\
\text{else} \\
& h_d := h_{sd} \\
& x := x + \alpha h_d
\end{align*}
\]
2-phase methods: General Algorithm Framework

**Algo#1**: 2-phase Descent Method (a general framework)

begin

\[ k := 0; \ x := x_0; \ found := \text{false} \]  \hspace{1cm} \{\text{Starting point}\}

while (not found) and (k < k_{\text{max}})

\[ h_d := \text{search\_direction}(x) \]  \hspace{1cm} \{\text{From } x \text{ and downhill}\}

if (no such \( h \) exists)

\[ found := \text{true} \]  \hspace{1cm} \{x \text{ is stationary}\}

else

\[ \alpha := \text{step\_length}(x, h_d) \]  \hspace{1cm} \{from \( x \) in direction \( h_d \}\}

\[ x := x + \alpha h_d; \quad k := k + 1 \]  \hspace{1cm} \{next iterate\}

end
2-phase methods: Line search to find the step length

Given a point \( x \) and a descent direction \( h \). The next iteration step is a move from \( x \) in direction \( h \). To find out, how far to move, we study the variation of the given function along the half line from \( x \) in the direction \( h \),

\[
\phi(\alpha) = F(x + \alpha h), \quad x \text{ and } h \text{ are fixed, } \alpha \geq 0
\]

Since \( h \) is a descent direction, when \( \alpha \) is small \( \phi(\alpha) < \phi(0) \)

An example of the behavior of \( \phi(\alpha) \),

![Variation of the function value along the search line](image_url)
2-phase methods: Line search to find the step length

• Line search to determine $\alpha$
  – $\alpha$ is iterated from an initial guess, e.g., $\alpha = 1$, then three different situations can arise
    1. $\alpha$ is so small that the gain in value of the function is very small; $\alpha$ should be increased
    2. $\alpha$ is too large: $\phi(\alpha) \geq \phi(0)$
       $\alpha$ should be decreased to satisfy the descent condition
    3. $\alpha$ is close to the minimizer of $\phi(\alpha)$. Accept this $\alpha$ value
Descent Methods

2-phase methods
(direction and step length are determined in 2 phases separately)

Phase I
Methods for computing descent direction
✓ Steepest descent method
✓ Newton’s method
✓ SD and Newton hybrid

Phase II
Methods for computing the step length
✓ Line search

1-phase methods
(direction and step length are determined jointly)
✓ Trust region methods
✓ Damped methods
  • Ex: Damped Newton method
1-phase methods: approximation model for $F$

Both trust region and damped methods assume that we have a model $L$ of the behavior of $F$ in the neighborhood of the current iterate $x$,

$$F(x + h) \approx L(h) = F(x) + h^T c + \frac{1}{2} h^T Bh$$

where $c \in \mathbb{R}^n$ and $B \in \mathbb{R}^{n \times n}$ is symmetric

For example, the model can be a second order Taylor expansion of $F$ around $x$
1-phase methods: trust region method

In a **trust region method** we assume that we know a positive number $\Delta$ such that the model is sufficiently accurate inside a ball with radius $\Delta$, centered at $x$, and determine the step as

$$h = h_{tr} \equiv \arg \min_{\|h\| \leq \Delta} \{ L(h) \}$$

$$h_{tr} = \arg \min_h L(h), \ s.t., \ h^T h \leq \Delta^2 \text{ (Eq. 1)}$$

**Note that: $h_{tr}$ consists of two parts of information, the direction and the step length**

So, basic steps to update using a trust region method are,

- compute $h$ by (1)
- if $F(x+h) < F(x)$
  - $x := x + h$
- update $\Delta$

Usually, we do not need to solve Eq. (1); instead, we can compute $h_{tr}$ in an approximation way, such as Dog Leg method.
1-phase methods: trust region method

- For each iteration, we modify $\Delta$
  - If the step fails, the reason is $\Delta$ is too large, and should be reduced
  - If the step is accepted, it may be possible to use a larger step from the new iterate

- The quality of the model with the computed step can be evaluated by the gain ratio,

**Definition 4**: Gain ratio

$$\rho = \frac{F(x) - F(x+h)}{L(0) - L(h)}$$

- the actual decrease
- the predicted decrease

This part is constructed to be positive. Why?
1-phase methods: trust region method

• If $\rho$ is small, indicating that the step is too large
• If $\rho$ is large, meaning that the approximation of $L$ to $F$ is good and we can try an even larger step

**Algo#2** The updating strategy for trust region radius $\Delta$

if $\rho < 0.25$
   $\Delta := \Delta / 2$
elseif $\rho > 0.75$
   $\Delta := \max \{\Delta, 3 \times \|h\|\}$
Descent Methods

2-phase methods
(direction and step length are determined in 2 phases separately)

1-phase methods
(direction and step length are determined jointly)

☑ Trust region methods
☑ Damped methods
  • Ex: Damped Newton method

Methods for computing descent direction
☑ Steepest descent method
☑ Newton’s method
☑ SD and Newton hybrid

Methods for computing the step length
☑ Line search
1-phase methods: damped method

In a damped method the step is determined as,

\[ h = h_{dm} \equiv \arg \min_h \left\{ L(h) + \frac{1}{2} \mu h^T h \right\} \quad \text{(Eq. 2)} \]

where \( \mu \geq 0 \) is the damping parameter. The term \( \frac{1}{2} \mu h^T h \) is used to penalize large steps.

The step \( h_{dm} \) is computed as a stationary point for the function,

\[ \phi_\mu(h) = L(h) + \frac{1}{2} \mu h^T h \]

Indicating that \( h_{dm} \) is a solution to,

\[ \phi_\mu'(h) = 0 \]
1-phase methods: damped method

\[ \phi'_\mu(h) = \frac{d}{dh} \left( L(h) + \frac{1}{2} \mu h^T h \right) = \frac{d}{dh} \left( F(x) + h^T c + \frac{1}{2} h^T B h + \frac{1}{2} \mu h^T h \right) \]

\[ = c + \frac{1}{2} \left( B + B^T \right) h + \mu h = c + Bh + \mu h = 0 \]

\[ h_{dm} = -\left( B + \mu I \right)^{-1} c \quad \text{(Eq. 3)} \]
1-phase methods: damped method

So, basic steps to update using a damped method are (similar to the trust region method),

**Algo#3** Basic steps using a damped method

- compute \( h \) by (2)
- if \( F(x+h) < F(x) \)
  - \( x := x + h \)
- update \( \mu \)

*the core problem*
1-phase methods: damped method

- If \( \rho \) is small, we should increase \( \mu \) and thereby increase the penalty on large steps
- If \( \rho \) is large, indicating that \( L(h) \) is a good approximation to \( F(x+h) \) for the computed \( h \), and \( \mu \) may be reduced

<table>
<thead>
<tr>
<th>Algo#4</th>
<th>Algo#5</th>
</tr>
</thead>
<tbody>
<tr>
<td>The 1st updating strategy for ( \mu )</td>
<td>The 2nd updating strategy for ( \mu )</td>
</tr>
<tr>
<td><strong>if</strong> ( \rho &lt; 0.25 )</td>
<td>( v = 2 )</td>
</tr>
<tr>
<td>( \mu := \mu \times 2 )</td>
<td><strong>if</strong> ( \rho &gt; 0 )</td>
</tr>
<tr>
<td><strong>elseif</strong> ( \rho &gt; 0.75 )</td>
<td>( \mu := \mu \times \max \left{ \frac{1}{3}, 1-(2\rho-1)^3 \right} ); ( v := 2 )</td>
</tr>
<tr>
<td>( \mu := \mu / 3 )</td>
<td><strong>else</strong></td>
</tr>
<tr>
<td>(Marquart 1963)</td>
<td>( \mu := \mu \times v ); ( v := 2 \times v )</td>
</tr>
</tbody>
</table>
1-phase methods: damped method

Ex: Damped Newton method

\[ F(x + h) = L(h) = F(x) + h^T c + \frac{1}{2} h^T Bh \]

where \( c \in \mathbb{R}^n \) and \( B \in \mathbb{R}^{n \times n} \) is symmetric.

(Eq. 3) takes the form,

\[ h_{dn} = -\left(F''(x) + \mu I\right)^{-1} F'(x) \]

the so-called damped Newton step

If \( \mu \) is very large,

\[ h_{dn} \approx -\frac{1}{\mu} F'(x) \]

a short step in a direction close to the deepest descent direction

If \( \mu \) is very small,

\[ h_{dn} \approx -\left[F''(x)\right]^{-1} F'(x) \]

a step close to the Newton step

We can think of the damped Newton method as a hybrid between the steepest descent method and the Newton method.
Outline

• Why is least squares an important problem in autonomous driving?
• Linear Least Squares
• **Non-linear Least Squares**
  • General Methods for Non-linear Optimization
  • Non-linear Least Squares Problems
    • Basic Concepts
    • Gauss-Newton Method
    • Levenberg-Marquardt Method
    • Powell’s Dog Leg Method
Basic Concepts

- Formulation of non-linear least squares problems
  Given a vector function $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $m \geq n$
  We want to find,
  $$x^* = \arg \min_x \{F(x)\}$$
  where,
  $$F(x) = \frac{1}{2} \sum_{i=1}^{m} (f_i(x))^2 = \frac{1}{2} \|f(x)\|^2 = \frac{1}{2} f(x)^T f(x)$$

- Non-linear least squares problems can be solved by general optimization methods, which will have some specific forms in this special case
Basic Concepts

Taylor expansion for $f(x)$,

$$f(x+h) = \begin{bmatrix} f_1(x+h) \\ f_2(x+h) \\ \vdots \\ f_m(x+h) \end{bmatrix} = \begin{bmatrix} f_1(x) + (\nabla f_1(x))^T h + O(h^2) \\ f_2(x) + (\nabla f_2(x))^T h + O(h^2) \\ \vdots \\ f_m(x) + (\nabla f_m(x))^T h + O(h^2) \end{bmatrix} = f(x) + J(x)h + O(h^2) \quad (\text{Eq. 4})$$

$J(x) \in \mathbb{R}^{m \times n}$ is called the Jacobian matrix of $f(x)$.
Basic Concepts

\[ F(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{m} (f_i(\mathbf{x}))^2 = \frac{1}{2} \left[ f_1^2(\mathbf{x}) + f_2^2(\mathbf{x}) + \ldots + f_m^2(\mathbf{x}) \right] \]

\[ \frac{\partial F(\mathbf{x})}{\partial x_j} = \frac{1}{2} \frac{\partial}{\partial x_j} \left[ f_1^2(\mathbf{x}) + f_2^2(\mathbf{x}) + \ldots + f_m^2(\mathbf{x}) \right] \]

\[ = f_1(\mathbf{x}) \frac{\partial f_1(\mathbf{x})}{\partial x_j} + f_2(\mathbf{x}) \frac{\partial f_2(\mathbf{x})}{\partial x_j} + \ldots + f_m(\mathbf{x}) \frac{\partial f_m(\mathbf{x})}{\partial x_j} \]

\[ = \sum_{i=1}^{m} \left[ f_i(\mathbf{x}) \frac{\partial f_i(\mathbf{x})}{\partial x_j} \right] \]
Basic Concepts

\[ F'(x) = \begin{bmatrix} \frac{\partial F(x)}{\partial x_1} \\ \frac{\partial F(x)}{\partial x_2} \\ \vdots \\ \frac{\partial F(x)}{\partial x_n} \end{bmatrix} = \begin{bmatrix} f_1(x) \frac{\partial f_1}{\partial x_1} + f_2(x) \frac{\partial f_2}{\partial x_1} + \cdots + f_m(x) \frac{\partial f_m}{\partial x_1} \\ f_1(x) \frac{\partial f_1}{\partial x_2} + f_2(x) \frac{\partial f_2}{\partial x_2} + \cdots + f_m(x) \frac{\partial f_m}{\partial x_2} \\ \vdots \\ f_1(x) \frac{\partial f_1}{\partial x_n} + f_2(x) \frac{\partial f_2}{\partial x_n} + \cdots + f_m(x) \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{n \times m} \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{bmatrix} \]

\[ = (J(x))^T f(x) \]  

(Eq. 5)
Basic Concepts

\[
\frac{\partial F(x)}{\partial x_j} = \sum_{i=1}^{m} \left[ f_i(x) \frac{\partial f_i(x)}{\partial x_j} \right]
\]

\[
\frac{\partial^2 F(x)}{\partial x_j \partial x_k} = \sum_{i=1}^{m} \left[ \frac{\partial f_i(x)}{\partial x_j} \frac{\partial f_i(x)}{\partial x_k} + f_i(x) \frac{\partial^2 f_i(x)}{\partial x_j \partial x_k} \right]
\]

\[
F'(x) = (J(x))^T J(x) + \sum_{i=1}^{m} f_i(x) f_i'(x) \quad \text{(addition of a stack of matrices)}
\]

\[
\begin{align*}
\text{Dimensions:} & \quad n \times m \quad m \times n \quad 1 \times 1 \quad n \times n \\
\end{align*}
\]
Outline

• Why is least squares an important problem in autonomous driving?
• Linear Least Squares
• Non-linear Least Squares
  • General Methods for Non-linear Optimization
  • Non-linear Least Squares Problems
    • Basic Concepts
    • Gauss-Newton Method
    • Levenberg-Marquardt Method
    • Powell’s Dog Leg Method
Gauss-Newton Method

The Gauss-Newton method is based on a linear approximation to the components of $f$ (a linear model of $f$) in the neighborhood of $x$ (refer to Eq. 4),

$$f(x + h) = f(x) + J(x)h$$

We suppose $J$ has full column rank

$$F(x + h) \approx L(h) \equiv \frac{1}{2}(f(x + h))^T f(x + h) = \frac{1}{2} f^T f + h^T J^T f + \frac{1}{2} h^T J^T J h$$

The Gauss-Newton step $h_{gn}$ minimizes $L(h)$,

$$h_{gn} = \text{arg min}_h \{L(h)\}$$

$h_{gn}$ is the solution to,

$$\frac{dL(h)}{dh} = 0 \Rightarrow J^T f + \frac{1}{2} (J^T J + J^T J) h = 0$$

$$\Rightarrow h_{gn} = -(J^T J)^{-1} J^T f$$

It can be considered that the Gauss-Newton’s updating step is obtained by using the trust-region method with $\Delta=\text{inf}$, or by the damped method with $\mu=0$ (compare with Eq. 3)
Gauss-Newton Method

• Some notes about Gauss-Newton methods
  – The classical Gauss-Newton method uses $\alpha = 1$ in all steps, then it can be regarded as a 1-phase method)

We can use $h_{gn}$ for $h_d$ in Algo#1.

Solve

$$(J^T J) h_{gn} = -J^T f$$

$x := x + h_{gn}$
Gauss-Newton Method

- Some notes about Gauss-Newton methods
  - The **classical Gauss-Newton method** uses $\alpha = 1$ in all steps, then it can be regarded as a 1-phase method)
  - If $\alpha$ is elegantly searched by line search, it can be categorized as a 2-phase method

We can use $h_{gn}$ for $h_d$ in Algo#1.

Solve \[
\left(J^T J\right) h_{gn} = -J^T f
\]

$x := x + \alpha h_{gn}$

where $\alpha$ is obtained by line search
Gauss-Newton Method

• Some notes about Gauss-Newton methods
  – The classical Gauss-Newton method uses $\alpha = 1$ in all steps, then it can be regarded as a 1-phase method
  – If $\alpha$ is elegantly searched by line search, it can be categorized as a 2-phase method
  – For each iteration step, it requires that the Jacobian $J$ has full column rank

If $J$ has full column rank, $J^TJ$ is positive definite

Proof:

$J$ has full column rank $\Leftrightarrow$ $J$’s columns are linearly unrelated

\[ \forall x \neq 0, y = Jx \neq 0 \Rightarrow 0 < y^T y = (Jx)^T Jx = x^T J^T Jx \]

$J^TJ$ is positive definite
Outline

• Why is least squares an important problem in autonomous driving?
• Linear Least Squares

• Non-linear Least Squares
  • General Methods for Non-linear Optimization
  • Non-linear Least Squares Problems
    • Basic Concepts
    • Gauss-Newton Method
    • Levenberg-Marquardt Method
    • Powell’s Dog Leg Method
Levenberg-Marquardt Method

• L-M method can be considered as a *damped Gauss-Newton method*

Consider a linear approximation to the components of \( f \) (a linear model of \( f \)) in the neighborhood of \( x \),

\[
f(x + h) \approx f(x) + J(x)h
\]

We don’t require \( J \) has full column rank

\[
F(x + h) \approx L(h) \equiv \left( \frac{1}{2} f(x + h) \right)^T f(x + h) = \frac{1}{2} f^T f + h^T J^T f + \frac{1}{2} h^T J^T J h
\]

Based on damped method (refer to Eq. 2),

\[
h_{lm} = \arg \min_h L(h) + \frac{1}{2} \mu h^T h,
\]

where \( \mu > 0 \) is the damped coefficient. \( h_{lm} \) is the solution to,

\[
d \left( L(h) + \frac{1}{2} \mu h^T h \right)_{dh} = 0
\]

\[
h_{lm} = -\left( J^T J + \mu I \right)^{-1} J^T f
\]

**positive definite**
Let $A = J^T J$, then $A + \mu I$ is positive definite for $\mu > 0$

Proof:

$\forall x \neq 0$, $y = Jx$

$0 \leq y^T y = x^T J^T Jx = x^T Ax \Rightarrow A$ is positive semi-definite

All $A$’s eigen-values $\{\lambda_i \geq 0, i = 1, \ldots, N\}$

$Av_i = \lambda_i v_i$

$(A + \mu I)v_i = (\lambda_i + \mu)v_i$

I.e., all $(A + \mu I)$’s eigen-values $\{\lambda_i + \mu\} > 0$

$A + \mu I$ is positive definite
Levenberg-Marquardt Method

• L-M method can be considered as a *damped Gauss-Newton method*

L-M’s step:

\[ h_{lm} = - \left( J^T J + \mu I \right)^{-1} J^T f \]

Gauss-Newton’s step (if \( \alpha = 1 \)):

\[ h_{gn} = - \left( J^T J \right)^{-1} J^T f \]
Levenberg-Marquardt Method

- Updating strategy of $\mu$
  - $\mu$ influences both the direction and the size of the step, and this leads L-M without a specific line search
  - The initial $\mu$–value is related to the elements in $\left( J(x_0) \right)^T J(x_0)$ by letting,
    $$\mu_0 = \tau \cdot \max_i \left\{ \left( J^T J \right)_{ii}^{(0)} \right\}$$
  - During iteration, $\mu$ can be updated by Algo#4 or Algo#5
Levenberg-Marquardt Method

• Stopping criteria
  – For a minimizer $x^*$, ideally we will have $F'(x^*) = 0$
    So, we can use
    \[ \left\| F'(x) \right\|_\infty \leq \epsilon_1 \]
    as the first stopping criterion
  – If for the current iteration, the change of $x$ is too small,
    \[ \left\| x_{\text{new}} - x \right\|_2 \leq \epsilon_2 \left( \left\| x \right\|_2 + \epsilon_2 \right) \]
  – Finally, we need a safeguard against an infinite loop,
    \[ k \geq k_{\text{max}} \]
    where $k$ is the current iteration index
Levenberg-Marquardt Method

Algo#6: L-M Method

\begin{verbatim}
begin
k := 0; \nu := 2; \quad x := x_0
A := J(x)^T J(x); \quad g := J(x)^T f(x)
found := (\|g\|_\infty \leq \epsilon_1); \quad \mu := \tau \cdot \max\{a_{ii}\}
while (not found) and (k < k_{max})
    k := k + 1;
    Solve \((A + \mu I)h_{lm} = -g\)
    if \|h_{lm}\| \leq \epsilon_2 (\|x\| + \epsilon_2)
        found := true
    else
        x_{new} := x + h_{lm}
        \rho := (F(x) - F(x_{new}))/((L(0) - L(h_{lm}))
        if \rho > 0
            \{step acceptable\}
            x := x_{new}
            A := J(x)^T J(x); \quad g := J(x)^T f(x)
            found := (\|g\|_\infty \leq \epsilon_1)
            \mu := \mu \cdot \max\{\frac{1}{3}, 1 - (2\rho - 1)^3\}; \quad \nu := 2
        else
            \mu := \mu \cdot \nu; \quad \nu := 2 \cdot \nu
end
\end{verbatim}

\textbf{g actually is} \(F'(x)\), see Eq. 5
Outline

• Why is least squares an important problem in autonomous driving?
• Linear Least Squares

• Non-linear Least Squares
  • General Methods for Non-linear Optimization
  • Non-linear Least Squares Problems
    • Basic Concepts
    • Gauss-Newton Method
    • Levenberg-Marquardt Method
    • Powell’s Dog Leg Method
Powell’s Dog Leg Method

- It works with combinations with the Gauss-Newton and the steepest descent directions
- It is a trust-region based method

Powell is a keen golfer!

Michael James David Powell (29 July 1936 – 19 April 2015) was a British mathematician, who worked at the University of Cambridge
Powell’s Dog Leg Method

Gauss-Newton step

$$h_{gn} = -(J^T J)^{-1} J^T f$$

The steepest descent direction

$$h_{sd} = -F'(x) = -(J(x))^T f(x)$$

This is the direction, not a step, and to see how far we should go, we look at the linear model,

$$f(x + \alpha h_{sd}) \approx f(x) + \alpha J(x)h_{sd}$$

$$F(x + \alpha h_{sd}) \approx \frac{1}{2} \left\| f(x) + \alpha J(x)h_{sd} \right\|^2 = F(x) + \alpha h_{sd}^T (J(x))^T f(x) + \frac{1}{2} \alpha^2 h_{sd}^T (J(x))^T J(x)h_{sd}$$

This function of $\alpha$ is minimal for,

$$\alpha = \frac{-h_{sd}^T J^T f}{h_{sd}^T J^T Jh_{sd}} = \frac{F'(x)^T F'(x)}{h_{sd}^T J^T Jh_{sd}} = \frac{\left\| F'(x) \right\|^2}{h_{sd}^T J^T Jh_{sd}}$$

(Eq. 6)
Powell’s Dog Leg Method

Now, we have two candidates for the step to take from the current point \( x \),

\[
a = \alpha h_{sd},\ b = h_{gn}
\]

Powell suggested to use the following strategy for choosing the step, when the trust region has the radius \( \Delta \)

```
Algo#6

if \( \| h_{gn} \| \leq \Delta \)
   \( h_{dl} := h_{gn} \)
elseif \( \| \alpha h_{sd} \| \geq \Delta \)
   \( h_{dl} := \frac{\Delta}{\| h_{sd} \|} h_{sd} \)
else
   \( h_{dl} := \alpha h_{sd} + \beta (h_{gn} - \alpha h_{sd}) \)
with chosen \( \beta \) so that \( \| h_{dl} \| = \Delta \)
```

```
Powell’s Dog Leg Method

The name *Dog Leg* is taken from golf: The fairway at a “dog leg hole” has a shape as the line from \( x \) (the tee point) via the end point of \( a \) to the end point of \( h_{dl} \) (the hole).
Powell’s Dog Leg Method

Algo#7: Dog Leg Method

begin
  \( k := 0; \ x := x_0; \ \Delta := \Delta_0; \ g := J(x)^	op f(x) \)
  \( \text{found := (} \|f(x)\|_\infty \leq \varepsilon_3 \text{)} \text{ or } (\|g\|_\infty \leq \varepsilon_1) \)
  while (not found) and (\( k < k_{\text{max}} \))
    \( k := k+1; \ \text{Compute } \alpha \text{ by } (\text{Eq. } 6) \)
    \( h_{sd} := -\alpha g; \ \text{Solve } J(x)h_{gn} \simeq -f(x) \)
    Compute \( h_{dl} \) by (Algo# 6)
    if \( \|h_{dl}\| \leq \varepsilon_2(\|x\| + \varepsilon_2) \)
      \( \text{found := true} \)
    else
      \( x_{\text{new}} := x + h_{dl} \)
      \( \beta := (F(x) - F(x_{\text{new}}))/(L(0) - L(h_{dl})) \)
      if \( \beta > 0 \)
        \( x := x_{\text{new}}; \ g := J(x)^	op f(x) \)
        \( \text{found := (} \|f(x)\|_\infty \leq \varepsilon_3 \text{)} \text{ or } (\|g\|_\infty \leq \varepsilon_1) \)
      if \( \beta > 0.75 \)
        \( \Delta := \max\{\Delta, 3\|h_{dl}\|\} \)
      elseif \( \beta < 0.25 \)
        \( \Delta := \Delta/2; \ \text{found := (} \Delta \leq \varepsilon_2(\|x\| + \varepsilon_2) \text{)} \)
end