Lecture 7
Least Squares

Lin ZHANG, PhD
School of Software Engineering
Tongji University
Fall 2023

Outline

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

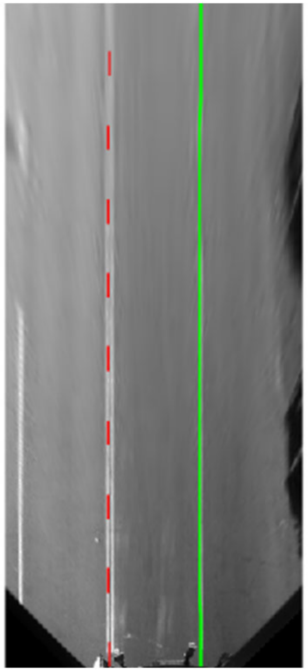
Why is least squares an important problem?

In intelligent automobile industry, some mathematical terminologies are often met

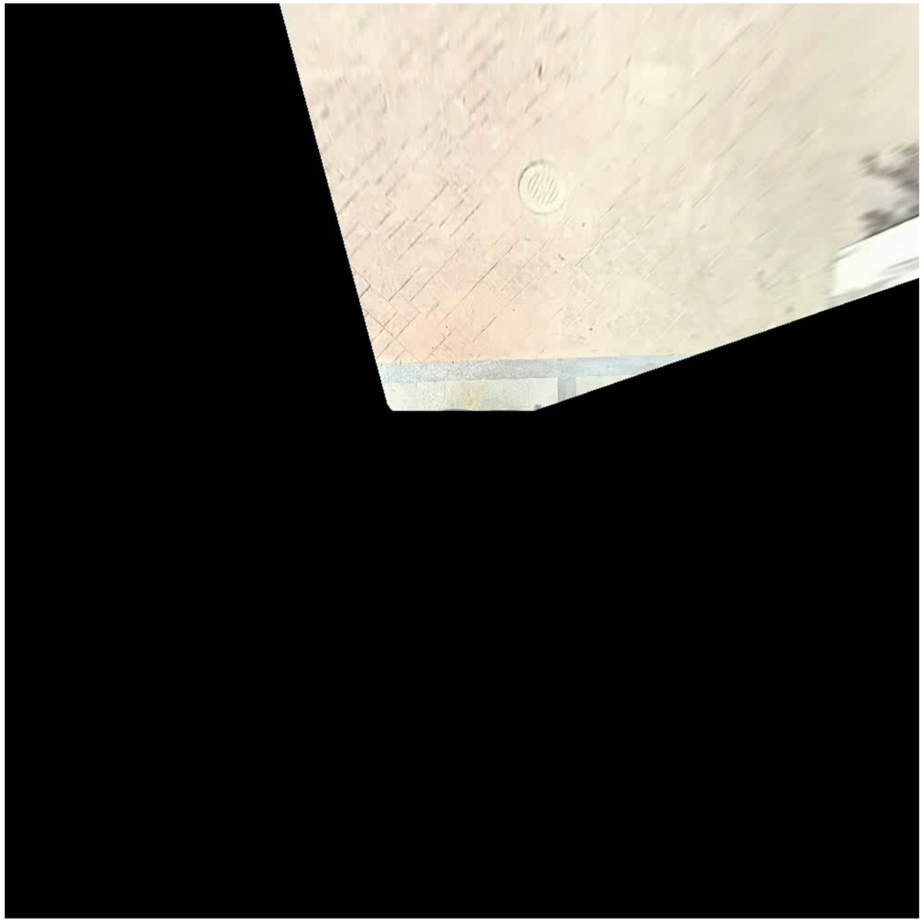
- Hessian matrix
- Jacobian matrix
- Damped method
- Homogeneous linear equation system
- Trust-region method
- Damped Newton method
- Inhomogeneous linear equation system
- Gauss-Newton method
- Levenberg-Marquardt method
- Lagrange multiplier
- Line search
- Steepest descent method
- Newton method
- 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.

$$\left\{ \mathbf{x}_i \leftrightarrow \mathbf{u}_i \right\}_{i=1}^{n}$$

A point on the physical plane   A point on the image plane

We know there existing an **H** satisfying

$$\mathbf{x}_i = \mathbf{H}\mathbf{u}_i$$

We need to find **H** from $\left\{ \mathbf{x}_i \leftrightarrow \mathbf{u}_i \right\}_{i=1}^{n}$

Why is least squares an important problem?

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

$$s\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \Rightarrow \begin{cases} h_{11}u + h_{12}v + h_{13} = sx \\ h_{21}u + h_{22}v + h_{23} = sy \\ h_{31}u + h_{32}v + h_{33} = s \end{cases} \Rightarrow \begin{cases} \dfrac{h_{11}u + h_{12}v + h_{13}}{h_{31}u + h_{32}v + h_{33}} = x \\ \dfrac{h_{21}u + h_{22}v + h_{23}}{h_{31}u + h_{32}v + h_{33}} = y \end{cases}$$

$$\Rightarrow \begin{pmatrix} u & v & 1 & 0 & 0 & 0 & -ux & -vx & -x \\ 0 & 0 & 0 & u & v & 1 & -uy & -vy & -y \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = \mathbf{0}$$

$\Rightarrow$ Since we have n point pairs, we get

$$\mathbf{A}_{2n\times 9}\mathbf{h}_{9\times 1} = \mathbf{0}$$

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,

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

$$\Rightarrow \begin{pmatrix} u & v & 1 & 0 & 0 & 0 & -ux & -vx \\ 0 & 0 & 0 & u & v & 1 & -uy & -vy \end{pmatrix}\begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{pmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$

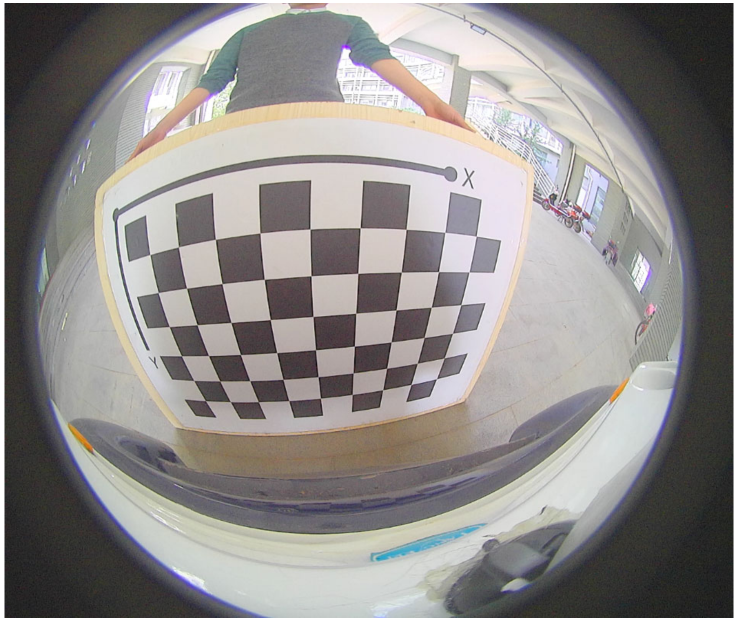
$\Rightarrow$ Since we have n point pairs, we get

$$\mathbf{A}_{2n\times 8}\mathbf{h}_{8\times 1} = \mathbf{b}_{2n\times 1}$$

How to solve this ***inhomogeneous*** linear equation system?

Why is least squares an important problem?

Ex2: Camera calibration



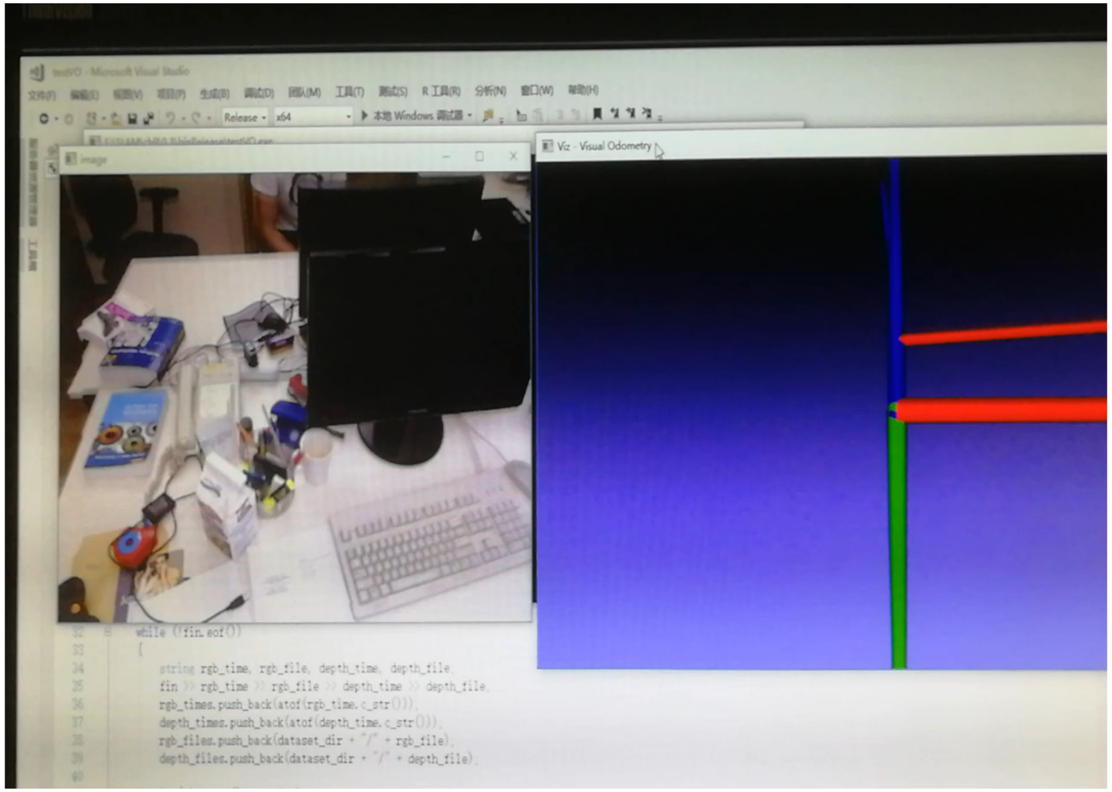
The widely used Zhang Zhengyou's method actually needs to solve a non-linear minimization problem,

$$\mathbf{A}^*, \mathbf{R}_i^*, \mathbf{t}_i^* = \arg\min_{\mathbf{A},\mathbf{R}_i,\mathbf{t}_i} \sum_{i=1}^{n} \sum_{j=1}^{m} \left\| \mathbf{m}_{ij} - \widehat{\mathbf{m}}\left(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{M}_j\right) \right\|^2$$

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,

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

where $\mathbf{p}_i$ is a 3D feature point in the world coordinate system, $\mathbf{u}_i$ is $\mathbf{p}_i$'s projection on the current frame, and $\mathbf{K}$ is the intrinsic matrix of the camera. We need to identify the optimal pose ξ^* 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

$$\mathbf{Ax} = \mathbf{b},\ \mathbf{b} \neq \mathbf{0},\ \mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{x} \in \mathbb{R}^{n \times 1}$$

Homogeneous linear equation system

$$\mathbf{Ax} = \mathbf{0},\ \mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{x} \in \mathbb{R}^{n \times 1}$$

Linear least squares

Non-linear least squares problem

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \frac{1}{2} \left\| \mathbf{f}(\mathbf{x}) \right\|^2$$

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

LS for Inhomogeneous Linear System

$$\mathbf{Ax} = \mathbf{b}, \mathbf{b} \neq \mathbf{0}, \mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{x} \in \mathbb{R}^{n \times 1}$$

The solution of the above problem can fall into three situations,

1) It has a unique solution
2) It has infinite solutions
3) It has no solution

What are conditions for these three cases?

We can solve the following linear least square problem to deal with all aforementioned three cases uniformly,

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \left\| \mathbf{Ax} - \mathbf{b} \right\|_2^2$$

LS for Inhomogeneous Linear System

Linear least squares is a general idea for solving linear equations,

$$\mathbf{A}_{m\times n}\mathbf{x}_{n\times 1}=\mathbf{b}_{m\times 1} \tag{1}$$

Using the idea of least squares, Eq. 1 is equivalent to the following problem,

$$\mathbf{x}^{*}=\arg\min_{\mathbf{x}}\left\|\mathbf{A}_{m\times n}\mathbf{x}_{n\times 1}-\mathbf{b}_{m\times 1}\right\|_2^2 \quad \textbf{(convex)} \tag{2}$$

Eq. 2 can be solved by finding the stationary point $\mathbf{x}^*$ of $\left\|\mathbf{A}_{m\times n}\mathbf{x}_{n\times 1}-\mathbf{b}_{m\times 1}\right\|_2^2$, i.e. $\mathbf{x}^*$ should satisfy,

$$\mathbf{A}^T\mathbf{A}\mathbf{x}^*=\mathbf{A}^T\mathbf{b} \tag{3}$$

In Eq. 3, when $rank(\mathbf{A})=n$ (the columns of $\mathbf{A}$ are linearly independent) ,

$rank(\mathbf{A}^T\mathbf{A})=n$ ➡ $\mathbf{A}^T\mathbf{A}$ is invertible ➡ $\mathbf{x}^*$ is uniquely determined as $\mathbf{x}^*=(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$

How about when rank(**A**)<n?

LS for Inhomogeneous Linear System

- For solving the linear least squares numerically with a computer, usually we do not use the form of Eq. (3) (though it is elegant) for two reasons
  - When rank($\mathbf{A}$)<n, $\mathbf{x}^*$ can not be determined
  - Even though $\mathbf{A}^T\mathbf{A}$ is invertible, the formation of $\mathbf{A}^T\mathbf{A}$ can dramatically degrade the accuracy of the computation
- Instead, we can use the technique of SVD

LS for Inhomogeneous Linear System

Suppose the SVD form of A is,

$$\mathbf{A}_{m\times n}=U_{m\times m}\Sigma_{m\times n}V_{n\times n}^{T}$$

$$\Downarrow$$

$$\mathbf{Ax}-\mathbf{b}=U\Sigma V^{T}\mathbf{x}-\mathbf{b}=U\left(\Sigma V^{T}\mathbf{x}\right)-U\left(U^{T}\mathbf{b}\right)\triangleq U\left(\Sigma\mathbf{y}_{n\times 1}-\mathbf{c}_{m\times 1}\right)$$

where $\mathbf{y}_{n\times 1}=V^{T}\mathbf{x},\mathbf{c}_{m\times 1}=U^{T}\mathbf{b}$

Since U is an orthogonal matrix,

$$\|\mathbf{Ax}-\mathbf{b}\|=\left\|U\left(\Sigma\mathbf{y}_{n\times 1}-\mathbf{c}_{m\times 1}\right)\right\|=\left\|\Sigma\mathbf{y}_{n\times 1}-\mathbf{c}_{m\times 1}\right\|$$

Then, our objective is to identify $\mathbf{y}$ that can make $\left\|\Sigma\mathbf{y}_{n\times 1}-\mathbf{c}_{m\times 1}\right\|$ have minimum length

LS for Inhomogeneous Linear System

$$\Sigma \mathbf{y}_{n\times 1} = \begin{bmatrix} \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix} & \boldsymbol{O}_{r\times(n-r)} \\ \boldsymbol{O}_{(m-r)\times r} & \boldsymbol{O}_{(m-r)\times(n-r)} \end{bmatrix}_{m\times n} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \sigma_1 y_1 \\ \sigma_2 y_2 \\ \vdots \\ \sigma_r y_r \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{m\times 1} \Rightarrow \Sigma \mathbf{y}_{n\times 1} - \mathbf{c}_{m\times 1} = \begin{bmatrix} \sigma_1 y_1 - c_1 \\ \sigma_2 y_2 - c_2 \\ \vdots \\ \sigma_r y_r - c_r \\ -c_{r+1} \\ \vdots \\ -c_m \end{bmatrix}_{m\times 1}$$

Then, we simply let $y_i = \frac{c_i}{\sigma_i}, 1 \le i \le r$; then, $\left\| \Sigma \mathbf{y}_{n\times 1} - \mathbf{c}_{m\times 1} \right\|$ can get the minimum length $\sqrt{\sum_{i=r+1}^{m} c_i^2}$

Note that $y_{r+1} \sim y_n$ can be arbitrary

LS for Inhomogeneous Linear System

The operation $y_i = \dfrac{c_i}{\sigma_i}, 1 \le i \le r$ can be simply completed by a matrix multiplication,

$$\mathbf{y} = \begin{bmatrix} \begin{bmatrix} \frac{1}{\sigma_1} & & & \\ & \frac{1}{\sigma_2} & & \\ & & \ddots & \\ & & & \frac{1}{\sigma_r} \end{bmatrix} & \boldsymbol{O}_{r\times(m-r)} \\ \boldsymbol{O}_{(n-r)\times r} & \boldsymbol{O}_{(n-r)\times(m-r)} \end{bmatrix}_{n\times m} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}_{m\times 1} = \begin{bmatrix} c_1/\sigma_1 \\ c_2/\sigma_2 \\ \vdots \\ c_r/\sigma_r \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n\times 1} \triangleq \Sigma^+ \mathbf{c}_{m\times 1}$$

where Σ^+ means transposing Σ and inverting all non-zero diagonal entries

Finally,

$$\mathbf{x} = V\mathbf{y}_{n\times 1} = V\Sigma^+\mathbf{c}_{m\times 1} = V\Sigma^+U^T\mathbf{b}$$

Moore-Penrose inverse

LS for Inhomogeneous Linear System

- Some notes about the generalized inverse used in linear least squares
  - It does not have requirements for the rank of $\mathbf{A}$
  - It can guarantee that the obtained solution can make $\|\mathbf{Ax}-\mathbf{b}\|$ having the minimum length; but **the solution may be not unique**

LS for Homogeneous Linear System

Consider the following homogeneous linear equations,

$$\mathbf{A}_{m\times n}\mathbf{x}_{n\times 1} = \mathbf{0}$$

The solution of the above problem can fall into two situations,

1) It only has the solution zero
2) It has both zero and non-zero solutions

In most cases, the trivial solution $\mathbf{x}$ = 0 has no use and thus we add a constraint $\|\mathbf{x}\|_2 = 1$ (actually 1 can be any other integer)

We can solve the following linear least square problem to deal with all aforementioned two cases uniformly,

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x}\|_2^2, \; s.t., \; \|\mathbf{x}\|_2 = 1$$

LS for Homogeneous Linear System

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x}\|_2^2, \ s.t., \ \|\mathbf{x}\|_2 = 1$$

Use the Lagrange multiplier to solve it,

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \left[\|\mathbf{A}\mathbf{x}\|_2^2 + \lambda\left(1 - \|\mathbf{x}\|_2^2\right) \right]$$

Solving the stationary point of the Lagrange function,

$$\begin{cases} \dfrac{\partial \left[\|\mathbf{A}\mathbf{x}\|_2^2 + \lambda\left(1 - \|\mathbf{x}\|_2^2\right) \right]}{\partial \mathbf{x}} = \mathbf{0} \\ \dfrac{\partial \left[\|\mathbf{A}\mathbf{x}\|_2^2 + \lambda\left(1 - \|\mathbf{x}\|_2^2\right) \right]}{\partial \lambda} = 0 \end{cases}$$

LS for Homogeneous Linear System

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

Then, we have

$$\mathbf{A}^T\mathbf{A}\mathbf{x}=\lambda\mathbf{x}$$

$\mathbf{x}$ is the eigen-vector of $\mathbf{A}^T\mathbf{A}$ associated with the eigenvalue λ

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

The unit vector $\mathbf{x}$ is the eigenvector associated with the minimum eigenvalue of $\mathbf{A}^T\mathbf{A}$

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

Basic Concepts

Definition 1: Local minimizer

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

$$F\left(\mathbf{x}^*\right) \le F(\mathbf{x}), \text{ for } \left\|\mathbf{x} - \mathbf{x}^*\right\| < \delta$$

where δ is a small positive number

Basic Concepts

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

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

where $\mathbf{F}'(\mathbf{x})$ is the gradient and $\mathbf{F}''(\mathbf{x})$ is the Hessian,

$$\mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F}{\partial x_1}(\mathbf{x}) \\ \frac{\partial F}{\partial x_2}(\mathbf{x}) \\ \vdots \\ \frac{\partial F}{\partial x_n}(\mathbf{x}) \end{bmatrix}, \quad \mathbf{F}''(\mathbf{x}) = \left[\frac{\partial^2 F}{\partial x_i \partial x_j}(\mathbf{x})\right]_{n\times n} = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1 \partial x_1} & \frac{\partial^2 F}{\partial x_1 \partial x_2} \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1} & \frac{\partial^2 F}{\partial x_2 \partial x_2} \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n} \\ \vdots & & \\ \frac{\partial^2 F}{\partial x_n \partial x_1} & \frac{\partial^2 F}{\partial x_n \partial x_2} \cdots & \frac{\partial^2 F}{\partial x_n \partial x_n} \end{bmatrix}_{n\times n}$$

Basic Concepts

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

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

where $\mathbf{F}'(\mathbf{x})$ is the gradient and $\mathbf{F}''(\mathbf{x})$ is the Hessian,

It is easy to verify that,

$$\mathbf{F}''(\mathbf{x}) = \frac{d\mathbf{F}'(\mathbf{x})}{d\mathbf{x}^T}$$

Basic Concepts

Theorem 1: Necessary condition for a local minimizer

If $\mathbf{x}^*$ is a local minimizer, then

$$\mathbf{F}'\left(\mathbf{x}^*\right) = \mathbf{0}$$

Definition 2: Stationary point

If $\mathbf{F}'\left(\mathbf{x}_s\right) = \mathbf{0}$,

then $\mathbf{x}_s$ 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**

Basic Concepts

Theorem 2: Sufficient condition for a local minimizer

Assume that $\mathbf{x}_s$ is a stationary point and that $\mathbf{F}''(\mathbf{x}_s)$ is positive definite, then $\mathbf{x}_s$ is a local minimizer

If $\mathbf{F}''(\mathbf{x}_s)$ is negative definite, then $\mathbf{x}_s$ is a local maximizer. If $\mathbf{F}''(\mathbf{x}_s)$ is indefinite (ie. it has both positive and negative eigenvalues), then $\mathbf{x}_s$ 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 starting point $\mathbf{x}_0$ the method produces a series of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots,$ which (hopefully) converges to $\mathbf{x}^*$
- The methods have measures to enforce the descending condition,

$$F(\mathbf{x}_{k+1}) < F(\mathbf{x}_k)$$

  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

Descent Methods

Consider the variation of the F-value along the half line starting at $\mathbf{x}$ and with direction $\mathbf{h}$,

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

Definition 3: Descent direction

h is a descent direction for F at $\mathbf{x}$ if

$$\mathbf{h}^T\mathbf{F}'(\mathbf{x}) < 0$$

Descent Methods

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
  $k := 0; \ \ \mathbf{x} := \mathbf{x}_0; \ \ \mathit{found} := \mathbf{false}$ {Starting point}
  while $(\textbf{not}\ \mathit{found})$ **and** $(k < k_{\max})$
    $\boxed{\mathbf{h}_\mathrm{d} := \mathrm{search\_direction}(\mathbf{x})}$ {From x and downhill}
    if (no such h exists)
      $\mathit{found} := \mathbf{true}$ {x is stationary}
    else
      $\alpha := \mathrm{step\_length}(\mathbf{x}, \mathbf{h}_\mathrm{d})$ {from x in direction $\mathbf{h}_\mathrm{d}$}
      $\mathbf{x} := \mathbf{x} + \alpha\mathbf{h}_\mathrm{d}; \quad k := k+1$ {next iterate}
end

2-phase methods: steepest descent to compute the descent direction

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

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

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

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

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 $\mathbf{h}_{sd}=-\mathbf{F}'(\mathbf{x})$

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 have 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 $\mathbf{x}^*$ is a stationary point, i.e.,

$$\mathbf{F}'\left(\mathbf{x}^*\right) = \mathbf{0}$$

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

$$\mathbf{F}'\left(\mathbf{x} + \mathbf{h}_n\right) = \mathbf{0}$$

what is the solution to $\mathbf{h}_n$?

2-phase methods: Newton's method to compute the descent direction

$$\mathbf{F}'(\mathbf{x}+\mathbf{h}) = \begin{bmatrix} \frac{\partial F}{\partial x_1}|_{\mathbf{x}+\mathbf{h}} \\ \frac{\partial F}{\partial x_2}|_{\mathbf{x}+\mathbf{h}} \\ \vdots \\ \frac{\partial F}{\partial x_n}|_{\mathbf{x}+\mathbf{h}} \end{bmatrix} \simeq \begin{bmatrix} \frac{\partial F}{\partial x_1}|_{\mathbf{x}} + \left(\nabla\left(\frac{\partial F}{\partial x_1}\right)|_{\mathbf{x}}\right)^T \mathbf{h} \\ \frac{\partial F}{\partial x_2}|_{\mathbf{x}} + \left(\nabla\left(\frac{\partial F}{\partial x_2}\right)|_{\mathbf{x}}\right)^T \mathbf{h} \\ \vdots \\ \frac{\partial F}{\partial x_n}|_{\mathbf{x}} + \left(\nabla\left(\frac{\partial F}{\partial x_n}\right)|_{\mathbf{x}}\right)^T \mathbf{h} \end{bmatrix} = \begin{bmatrix} \frac{\partial F}{\partial x_1}|_{\mathbf{x}} \\ \frac{\partial F}{\partial x_2}|_{\mathbf{x}} \\ \vdots \\ \frac{\partial F}{\partial x_n}|_{\mathbf{x}} \end{bmatrix} + \begin{bmatrix} \left(\nabla\left(\frac{\partial F}{\partial x_1}\right)|_{\mathbf{x}}\right)^T \mathbf{h} \\ \left(\nabla\left(\frac{\partial F}{\partial x_2}\right)|_{\mathbf{x}}\right)^T \mathbf{h} \\ \vdots \\ \left(\nabla\left(\frac{\partial F}{\partial x_n}\right)|_{\mathbf{x}}\right)^T \mathbf{h} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial F}{\partial x_1}|_{\mathbf{x}} \\ \frac{\partial F}{\partial x_2}|_{\mathbf{x}} \\ \vdots \\ \frac{\partial F}{\partial x_n}|_{\mathbf{x}} \end{bmatrix} + \begin{bmatrix} \frac{\partial^2 F}{\partial x_1 \partial x_1}|_{\mathbf{x}} & \frac{\partial^2 F}{\partial x_1 \partial x_2}|_{\mathbf{x}} & \cdots & \frac{\partial^2 F}{\partial x_1 \partial x_n}|_{\mathbf{x}} \\ \frac{\partial^2 F}{\partial x_2 \partial x_1}|_{\mathbf{x}} & \frac{\partial^2 F}{\partial x_2 \partial x_2}|_{\mathbf{x}} & \cdots & \frac{\partial^2 F}{\partial x_2 \partial x_n}|_{\mathbf{x}} \\ \vdots & & & \\ \frac{\partial^2 F}{\partial x_n \partial x_1}|_{\mathbf{x}} & \frac{\partial^2 F}{\partial x_n \partial x_2}|_{\mathbf{x}} & \cdots & \frac{\partial^2 F}{\partial x_n \partial x_n}|_{\mathbf{x}} \end{bmatrix} \mathbf{h} = \mathbf{F}'(\mathbf{x}) + \mathbf{F}''(\mathbf{x})\mathbf{h}$$

So $\mathbf{h}_n$ is the solution to,

$$\mathbf{F}''(\mathbf{x})\mathbf{h}_n = -\mathbf{F}'(\mathbf{x})$$

Suppose that $\mathbf{F}''(\mathbf{x})$ is positive definite, then,

$$\mathbf{h}_n^T \mathbf{F}''(\mathbf{x})\mathbf{h}_n = -\mathbf{h}_n^T \mathbf{F}'(\mathbf{x}) > 0$$

i.e.,

$$\mathbf{h}_n^T \mathbf{F}'(\mathbf{x}) < 0$$

indicates that $\mathbf{h}_n$ is a **descent direction**

In classical Newton method, the update is (then it can be regarded as a 1-phase method),

$$\mathbf{x} := \mathbf{x} + \mathbf{h}_n$$

However, in most modern implementations,

$$\mathbf{x} := \mathbf{x} + \alpha \mathbf{h}_n$$

where α 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 $\mathbf{x}$ is close to $\mathbf{x}^*$
  - Only when $\mathbf{F}''(\mathbf{x})$ is positive definite, it is sure that $\mathbf{h}_n$ 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

if $\mathbf{F}''(\mathbf{x})$ is positive definite

$\quad \mathbf{h}_d := \mathbf{h}_n$

else

$\quad \mathbf{h}_d := \mathbf{h}_{sd}$

$\mathbf{x} := \mathbf{x} + \alpha \mathbf{h}_d$

2-phase methods: General Algorithm Framework

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

begin
    $k := 0;\ \ \mathbf{x} := \mathbf{x}_0;\ \ \mathit{found} := \mathbf{false}$ $\{$Starting point$\}$
    while (**not** *found*) **and** $(k < k_{\max})$
        $\boxed{\mathbf{h}_{\mathrm{d}} := \mathrm{search\_direction}(\mathbf{x})}$ $\{$From $\mathbf{x}$ and downhill$\}$
        if (no such h exists)
            $\mathit{found} := \mathbf{true}$ $\{\mathbf{x}$ is stationary$\}$
        else
            $\alpha := \mathrm{step\_length}(\mathbf{x}, \mathbf{h}_{\mathrm{d}})$ $\{$from $\mathbf{x}$ in direction $\mathbf{h}_{\mathrm{d}}\}$
            $\mathbf{x} := \mathbf{x} + \alpha\mathbf{h}_{\mathrm{d}};\quad k := k+1$ $\{$next iterate$\}$
end

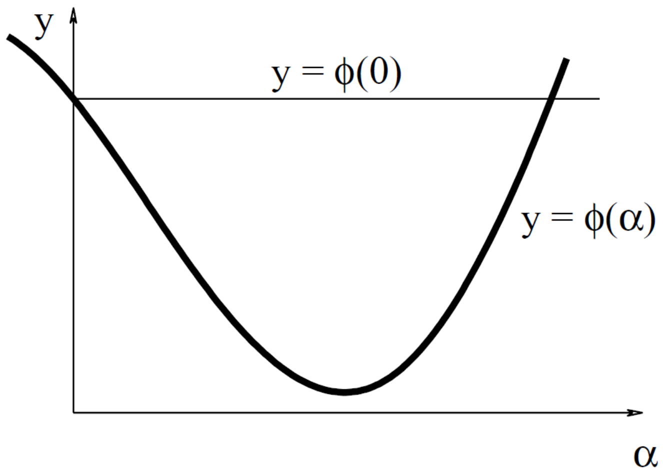
2-phase methods: Line search to find the step length

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

$$\phi(\alpha) = F(\mathbf{x} + \alpha\mathbf{h}),\ \mathbf{x} \text{ and } \mathbf{h} \text{ are fixed, } \alpha \geq 0$$

Since $\mathbf{h}$ is a descent direction, when α is small $\phi(\alpha) < \phi(0)$

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



Variation of the function value along the search line

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

- Line search to determine α
  - α is iterated from an initial guess, e.g., $\alpha = 1$, then three different situations can arise

    1. α is so small that the gain in value of the function is very small; α should be increased
    2. α is too large: $\phi(\alpha) \geq \phi(0)$ α should be decreased to satisfy the descent condition
    3. α is close to the minimizer of $\phi(\alpha)$. Accept this α value

Descent Methods

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 $\mathbf{x}$,

$$F(\mathbf{x}+\mathbf{h}) \simeq L(\mathbf{h}) = F(\mathbf{x}) + \mathbf{h}^T\mathbf{c} + \frac{1}{2}\mathbf{h}^T\mathbf{B}\mathbf{h}$$

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

For example, the model can be a second order Taylor expansion of F around $\mathbf{x}$

1-phase methods: trust region method

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

$$\mathbf{h} = \mathbf{h}_{tr} \equiv \arg\min_{\|\mathbf{h}\|\le\Delta} \left\{ L(\mathbf{h}) \right\}$$

$$\mathbf{h}_{tr} = \arg\min_{\mathbf{h}} L(\mathbf{h}),\ s.t.,\ \mathbf{h}^T\mathbf{h} \le \Delta^2 \quad \textbf{(Eq.1)}$$

Usually, we do not need to solve Eq. (1); instead, we can compute $\mathbf{h}_{tr}$ in an approximation way, such as Dog Leg method

Note that: $\mathbf{h}_{tr}$ 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 $F(\mathbf{x}+\mathbf{h})<F(\mathbf{x})$
    $\mathbf{x}:=\mathbf{x}+\mathbf{h}$
update Δ

the core problem

1-phase methods: trust region method

- For each iteration, we modify Δ
  - If the step fails, the reason is Δ 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

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

the actual decrease

the predicted decrease

This part is constructed be positive. Why?

1-phase methods: trust region method

- If ρ is small, indicating that the step is too large
- If ρ 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 Δ

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

Descent Methods

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: damped method

In a *damped method* the step is determined as,

$$\mathbf{h} = \mathbf{h}_{dm} \equiv \arg\min_{\mathbf{h}} \left\{ L(\mathbf{h}) + \frac{1}{2}\mu \mathbf{h}^T \mathbf{h} \right\} \qquad \textbf{(Eq. 2)}$$

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

The step $\mathbf{h}_{dm}$ is computed as a stationary point for the function,

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

Indicating that $\mathbf{h}_{dm}$ is a solution to,

$$\phi_\mu^{'}(\mathbf{h}) = 0$$

1-phase methods: damped method

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

$$= \mathbf{c} + \frac{1}{2}\left(\mathbf{B} + \mathbf{B}^T\right)\mathbf{h} + \mu\mathbf{h} = \mathbf{c} + \mathbf{B}\mathbf{h} + \mu\mathbf{h} = 0$$

$$\Rightarrow \quad \mathbf{h}_{dm} = -\left(\mathbf{B} + \mu\mathbf{I}\right)^{-1}\mathbf{c} \qquad \textbf{(Eq. 3)}$$

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 Eq. 2
if $F(\mathbf{x}+\mathbf{h})<F(\mathbf{x})$
    $\mathbf{x}:=\mathbf{x}+\mathbf{h}$
update μ

the core problem

1-phase methods: damped method

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

Algo#4

The 1st updating strategy for μ

$$
\begin{aligned}
&\textbf{if} \;\; \rho < 0.25 \\
&\quad \mu := \mu \times 2 \\
&\textbf{elseif} \;\; \rho > 0.75 \\
&\quad \mu := \mu / 3
\end{aligned}
$$

(Marquart 1963)

Algo#5

The 2nd updating strategy for μ

$$
\begin{aligned}
&v = 2 \\
&\textbf{if} \;\; \rho > 0 \\
&\quad \mu := \mu \times \max\left\{\frac{1}{3}, 1-(2\rho-1)^3\right\}; v := 2 \\
&\textbf{else} \\
&\quad \mu := \mu \times v; v := 2 \times v
\end{aligned}
$$

(Nielsen 1999)

1-phase methods: damped method

Ex: Damped Newton method

$$F(\mathbf{x}+\mathbf{h}) \simeq L(\mathbf{h}) = F(\mathbf{x}) + \mathbf{h}^T\mathbf{c} + \frac{1}{2}\mathbf{h}^T\mathbf{B}\mathbf{h}$$

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

if $\mathbf{c} = \mathbf{F}'(\mathbf{x})$ and $\mathbf{B} = \mathbf{F}''(\mathbf{x})$

(Eq. 3) takes the form,

$$\mathbf{h}_{dn} = -\left(\mathbf{F}''(\mathbf{x}) + \mu\mathbf{I}\right)^{-1}\mathbf{F}'(\mathbf{x})$$

the so-called damped Newton step

If μ is very large,

$$\mathbf{h}_{dn} \simeq -\frac{1}{\mu}\mathbf{F}'(\mathbf{x})$$, a short step in a direction close to the deepest descent direction

If μ is very small,

$$\mathbf{h}_{dn} \simeq -\left[\mathbf{F}''(\mathbf{x})\right]^{-1}\mathbf{F}'(\mathbf{x})$$, 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 $\mathbf{f}:\mathbb{R}^n \to \mathbb{R}^m, m \geq n$

We want to find,

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \{F(\mathbf{x})\}$$

where,

$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}\left(f_i(\mathbf{x})\right)^2 = \frac{1}{2}\left\|\mathbf{f}(\mathbf{x})\right\|^2 = \frac{1}{2}\mathbf{f}(\mathbf{x})^T\mathbf{f}(\mathbf{x})$$

- 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 $\mathbf{f}(\mathbf{x})$,

$$\mathbf{f}(\mathbf{x}+\mathbf{h}) = \begin{bmatrix} f_1(\mathbf{x}+\mathbf{h}) \\ f_2(\mathbf{x}+\mathbf{h}) \\ \vdots \\ f_m(\mathbf{x}+\mathbf{h}) \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{x}) + (\nabla f_1(\mathbf{x}))^T \mathbf{h} + O(\|\mathbf{h}\|^2) \\ f_2(\mathbf{x}) + (\nabla f_2(\mathbf{x}))^T \mathbf{h} + O(\|\mathbf{h}\|^2) \\ \vdots \\ f_m(\mathbf{x}) + (\nabla f_m(\mathbf{x}))^T \mathbf{h} + O(\|\mathbf{h}\|^2) \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} + \begin{bmatrix} (\nabla f_1(\mathbf{x}))^T \\ (\nabla f_2(\mathbf{x}))^T \\ \vdots \\ (\nabla f_m(\mathbf{x}))^T \end{bmatrix} \mathbf{h} + O(\|\mathbf{h}\|^2)$$

$$= \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|^2) \qquad \textbf{(Eq. 4)}$$

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

Basic Concepts

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

$$\Downarrow$$

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

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

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

Basic Concepts

$$\mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F(\mathbf{x})}{\partial x_1} \\ \frac{\partial F(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial F(\mathbf{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{x})\frac{\partial f_1}{\partial x_1} + f_2(\mathbf{x})\frac{\partial f_2}{\partial x_1} + \ldots + f_m(\mathbf{x})\frac{\partial f_m}{\partial x_1} \\ f_1(\mathbf{x})\frac{\partial f_1}{\partial x_2} + f_2(\mathbf{x})\frac{\partial f_2}{\partial x_2} + \ldots + f_m(\mathbf{x})\frac{\partial f_m}{\partial x_2} \\ \vdots \\ f_1(\mathbf{x})\frac{\partial f_1}{\partial x_n} + f_2(\mathbf{x})\frac{\partial f_2}{\partial x_n} + \ldots + f_m(\mathbf{x})\frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \ldots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \ldots & \frac{\partial f_m}{\partial x_2} \\ \vdots & & & \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \ldots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{n\times m} \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix}$$

$$= \left(\mathbf{J}(\mathbf{x})\right)^T \mathbf{f}(\mathbf{x}) \qquad \textbf{(Eq. 5)}$$

Basic Concepts

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

$$\Downarrow$$

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

$$\Downarrow$$

$$\mathbf{F}''(\mathbf{x}) = \underbrace{(\mathbf{J}(\mathbf{x}))^T}_{n\times m} \underbrace{\mathbf{J}(\mathbf{x})}_{m\times n} + \sum_{i=1}^{m} \underbrace{f_i(\mathbf{x})}_{1\times 1} \underbrace{\mathbf{f}_i''(\mathbf{x})}_{n\times n}$$

(addition of a stack of matrices)

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),

$$\mathbf{f}(\mathbf{x}+\mathbf{h}) \simeq \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h}$$

We suppose **J** has full column rank

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

The Gauss-Newton step $\mathbf{h}_{gn}$ minimizes $L(\mathbf{h})$,

$$\mathbf{h}_{gn} = \arg\min_{\mathbf{h}} \{L(\mathbf{h})\}$$

$\mathbf{h}_{gn}$ is the solution to,

$$\frac{dL(\mathbf{h})}{d\mathbf{h}} = \mathbf{0} \Rightarrow \mathbf{J}^T\mathbf{f} + \frac{1}{2}\left(\mathbf{J}^T\mathbf{J} + \mathbf{J}^T\mathbf{J}\right)\mathbf{h} = \mathbf{0}$$

$$\Rightarrow \mathbf{h}_{gn} = -\left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\mathbf{f}$$

It can be considered that the Gauss-Newton's updating step is obtained by using the trust-region method with Δ=inf, or by the damped method with μ=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 $\mathbf{h}_{gn}$ for $\mathbf{h}_d$ in **Algo#1**.

$$\text{Solve } \left(\mathbf{J}^T\mathbf{J}\right)\mathbf{h}_{gn} = -\mathbf{J}^T\mathbf{f}$$

$$\mathbf{x} := \mathbf{x} + \mathbf{h}_{gn}$$

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 α is elegantly searched by line search, it can be categorized as a 2-phase method

We can use $\mathbf{h}_{gn}$ for $\mathbf{h}_d$ in **Algo#1**.

$$\text{Solve } \left(\mathbf{J}^T\mathbf{J}\right)\mathbf{h}_{gn} = -\mathbf{J}^T\mathbf{f}$$

$$\mathbf{x} := \mathbf{x} + \alpha\mathbf{h}_{gn}$$

where α 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 α is elegantly searched by line search, it can be categorized as a 2-phase method
  - For each iteration step, it requires that the Jacobian $\mathbf{J}$ has full column rank

If $\mathbf{J}$ has full column rank, $\mathbf{J}^T\mathbf{J}$ is positive definite

Proof:

$\mathbf{J}$ has full column rank $\Leftrightarrow$ $\mathbf{J}$'s columns are linearly unrelated

$$\Downarrow$$

$$\forall \mathbf{x} \neq \mathbf{0}, \mathbf{y} = \mathbf{J}\mathbf{x} \neq \mathbf{0} \Rightarrow 0 < \mathbf{y}^T\mathbf{y} = \left(\mathbf{J}\mathbf{x}\right)^T \mathbf{J}\mathbf{x} = \mathbf{x}^T\mathbf{J}^T\mathbf{J}\mathbf{x}$$

$$\Downarrow$$

$\mathbf{J}^T\mathbf{J}$ 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**, $\mathbf{f}(\mathbf{x}+\mathbf{h}) \simeq \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h}$

We don't require **J** has full column rank

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

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

$$\mathbf{h}_{lm} = \arg\min_{\mathbf{h}} L(\mathbf{h}) + \frac{1}{2}\mu\mathbf{h}^T\mathbf{h}$$, where $\mu > 0$ is the damped coefficient

$\mathbf{h}_{lm}$ is the solution to,

$$\frac{d\left(L(\mathbf{h}) + \frac{1}{2}\mu\mathbf{h}^T\mathbf{h}\right)}{d\mathbf{h}} = 0 \Rightarrow \mathbf{h}_{lm} = -\left(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right)^{-1}\mathbf{J}^T\mathbf{f}$$

positive definite

Levenberg-Marquardt Method

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

Proof:

$$\forall \mathbf{x} \neq \mathbf{0}, \mathbf{y} = \mathbf{J}\mathbf{x}$$

$$0 \leq \mathbf{y}^T\mathbf{y} = \mathbf{x}^T\mathbf{J}^T\mathbf{J}\mathbf{x} = \mathbf{x}^T\mathbf{A}\mathbf{x} \Rightarrow \mathbf{A} \text{ is positive semi-definite}$$

$\Downarrow$

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

$$\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

$\Downarrow$

$$(\mathbf{A} + \mu\mathbf{I})\mathbf{v}_i = (\lambda_i + \mu)\mathbf{v}_i$$

$\Downarrow$

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

$\Downarrow$

$\mathbf{A} + \mu\mathbf{I}$ is positive definite

Levenberg-Marquardt Method

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

L-M's step:

$$\mathbf{h}_{lm} = -\left(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right)^{-1}\mathbf{J}^T\mathbf{f}$$

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

$$\mathbf{h}_{gn} = -\left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\mathbf{f}$$

That's why we say L-M is a damped Gauss-Newton method

Levenberg-Marquardt Method

- Updating strategy of μ
  - μ influences both the direction and the size of the step, and this leads L-M **without** a specific line search
  - The initial μ–value is related to the elements in $\left(\mathbf{J}(\mathbf{x_0})\right)^T \mathbf{J}(\mathbf{x_0})$ by letting,

$$\mu_0 = \tau \cdot \max_i \left\{ \left(\mathbf{J}^T \mathbf{J}\right)_{ii}^{(0)} \right\}$$

  - During iteration, μ can be updated by **Algo#4** or **Algo#5**

Levenberg-Marquardt Method

- Stopping criteria
  - For a minimizer $\mathbf{x}^*$, ideally we will have $\mathbf{F}'(\mathbf{x}^*) = 0$

    So, we can use

    $$\left\|\mathbf{F}'(\mathbf{x})\right\|_\infty \le \varepsilon_1$$

    as the first stopping criterion
  - If for the current iteration, the change of $\mathbf{x}$ is too small,

    $$\left\|\mathbf{x}_{new} - \mathbf{x}\right\|_2 \le \varepsilon_2 \left(\left\|\mathbf{x}\right\|_2 + \varepsilon_2\right)$$
  - Finally, we need a safeguard against an infinite loop,

    $$k \ge k_{\max}$$

    where k is the current iteration index

Levenberg-Marquardt Method

Algo#6: L-M Method

g actually is **F'(x)**, see Eq. 5

begin

$k := 0; \quad \nu := 2; \quad \mathbf{x} := \mathbf{x}_0$

$\mathbf{A} := \mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}); \quad \mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$

$found := (\|\mathbf{g}\|_\infty \le \varepsilon_1); \quad \mu := \tau * \max\{a_{ii}\}$

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

$k := k+1; \quad \text{Solve } (\mathbf{A} + \mu\mathbf{I})\mathbf{h}_{\text{lm}} = -\mathbf{g}$

if $\|\mathbf{h}_{\text{lm}}\| \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2)$

found := **true**

else

$\mathbf{x}_{\text{new}} := \mathbf{x} + \mathbf{h}_{\text{lm}}$

$\varrho := (F(\mathbf{x}) - F(\mathbf{x}_{\text{new}}))/(L(\mathbf{0}) - L(\mathbf{h}_{\text{lm}}))$

if $\varrho > 0$ {step acceptable}

$\mathbf{x} := \mathbf{x}_{\text{new}}$

$\mathbf{A} := \mathbf{J}(\mathbf{x})^\top \mathbf{J}(\mathbf{x}); \quad \mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$

$found := (\|\mathbf{g}\|_\infty \le \varepsilon_1)$

$\mu := \mu * \max\{\frac{1}{3}, 1 - (2\varrho - 1)^3\}; \quad \nu := 2$

else

$\mu := \mu * \nu; \quad \nu := 2 * \nu$

end

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** $\mathbf{h}_{gn} = -\left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T\mathbf{f}$

The steepest descent direction $\mathbf{h}_{sd} = -\mathbf{F}'(\mathbf{x}) = -\left(\mathbf{J}(\mathbf{x})\right)^T \mathbf{f}(\mathbf{x})$

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

$$\mathbf{f}(\mathbf{x}+\alpha\mathbf{h}_{sd}) \simeq \mathbf{f}(\mathbf{x}) + \alpha\mathbf{J}(\mathbf{x})\mathbf{h}_{sd}$$

$$F(\mathbf{x}+\alpha\mathbf{h}_{sd}) \simeq \frac{1}{2}\left\|\mathbf{f}(\mathbf{x}) + \alpha\mathbf{J}(\mathbf{x})\mathbf{h}_{sd}\right\|^2 = F(\mathbf{x}) + \alpha\mathbf{h}_{sd}^T\left(\mathbf{J}(\mathbf{x})\right)^T\mathbf{f}(\mathbf{x}) + \frac{1}{2}\alpha^2\mathbf{h}_{sd}^T\left(\mathbf{J}(\mathbf{x})\right)^T\mathbf{J}(\mathbf{x})\mathbf{h}_{sd}$$

This function of α is minimal for,

$$\alpha = \frac{-\mathbf{h}_{sd}^T\mathbf{J}^T\mathbf{f}}{\mathbf{h}_{sd}^T\mathbf{J}^T\mathbf{J}\mathbf{h}_{sd}} = \frac{\mathbf{F}'(\mathbf{x})^T\mathbf{F}'(\mathbf{x})}{\mathbf{h}_{sd}^T\mathbf{J}^T\mathbf{J}\mathbf{h}_{sd}} = \frac{\left\|\mathbf{F}'(\mathbf{x})\right\|^2}{\mathbf{h}_{sd}^T\mathbf{J}^T\mathbf{J}\mathbf{h}_{sd}} \quad \textbf{(Eq. 6)}$$

Powell’s Dog Leg Method

Now, we have two candidates for the step to take from the current point $\mathbf{x}$,

$$\mathbf{a} = \alpha \mathbf{h}_{sd}, \mathbf{b} = \mathbf{h}_{gn}$$

Powell suggested to use the following strategy for choosing the step, when the trust region has the radius Δ

Algo#6

if $\|\mathbf{h}_{gn}\| \le \Delta$

$\quad \mathbf{h}_{dl} := \mathbf{h}_{gn}$

elseif $\|\alpha \mathbf{h}_{sd}\| \ge \Delta$

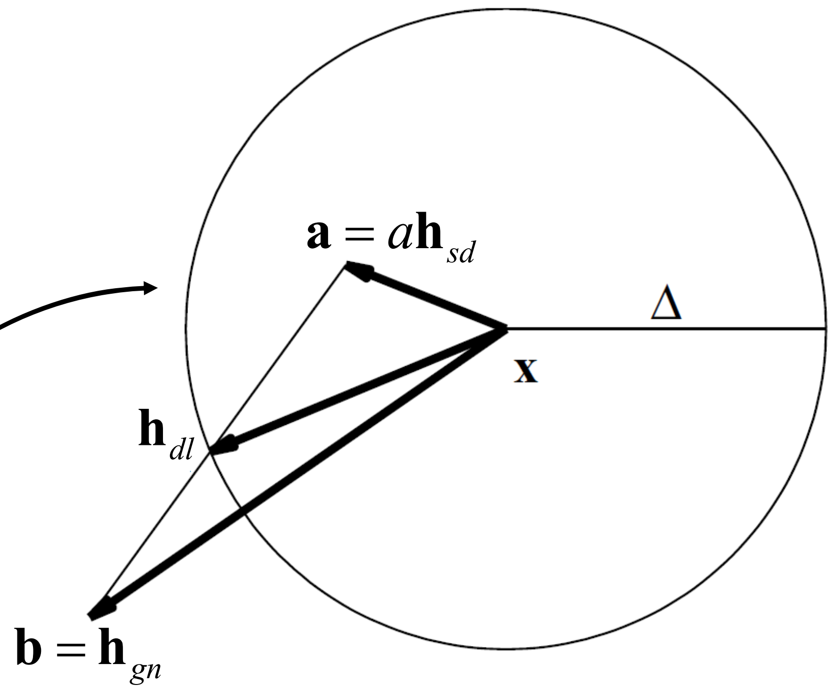
$\quad \mathbf{h}_{dl} := \frac{\Delta}{\|\mathbf{h}_{sd}\|} \mathbf{h}_{sd}$

else

$\quad \mathbf{h}_{dl} := \alpha \mathbf{h}_{sd} + \beta \left(\mathbf{h}_{gn} - \alpha \mathbf{h}_{sd} \right)$

$\quad$ with chosen β so that $\|\mathbf{h}_{dl}\| = \Delta$

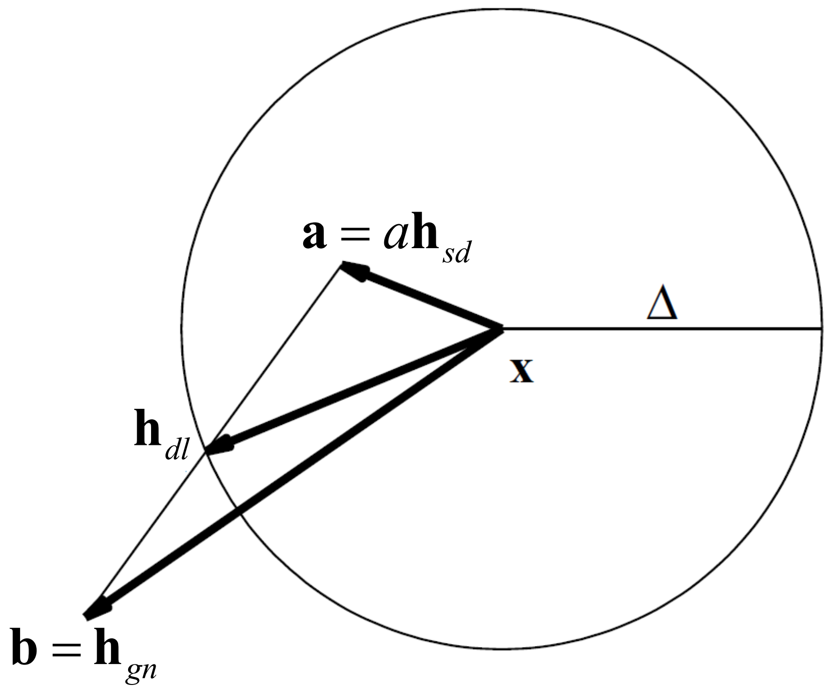
the last case



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 $\mathbf{h}_{dl}$ (the hole)

Dog Leg hole



Powell's Dog Leg Method

Algo#7: Dog Leg Method

begin
  $k := 0; \quad \mathbf{x} := \mathbf{x}_0; \quad \Delta := \Delta_0; \quad \mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$
  found $:= (\|\mathbf{f}(\mathbf{x})\|_\infty \le \varepsilon_3)$ **or** $(\|\mathbf{g}\|_\infty \le \varepsilon_1)$
  while (**not** *found*) **and** $(k < k_{\max})$
    $k := k+1$; Compute α by (Eq. 6)
    $\mathbf{h}_{\mathrm{sd}} := -\alpha \mathbf{g}$; Solve $\mathbf{J}(\mathbf{x})\mathbf{h}_{\mathrm{gn}} \simeq -\mathbf{f}(\mathbf{x})$
    Compute $\mathbf{h}_{\mathrm{dl}}$ by (Algo# 6)
    if $\|\mathbf{h}_{\mathrm{dl}}\| \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2)$
      found := **true**
    else
      $\mathbf{x}_{\mathrm{new}} := \mathbf{x} + \mathbf{h}_{\mathrm{dl}}$
      $\varrho := (F(\mathbf{x}) - F(\mathbf{x}_{\mathrm{new}}))/(L(\mathbf{0}) - L(\mathbf{h}_{\mathrm{dl}}))$
      if $\varrho > 0$
        $\mathbf{x} := \mathbf{x}_{\mathrm{new}}; \quad \mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$
        found $:= (\|\mathbf{f}(\mathbf{x})\|_\infty \le \varepsilon_3)$ **or** $(\|\mathbf{g}\|_\infty \le \varepsilon_1)$
      if $\varrho > 0.75$
        $\Delta := \max\{\Delta, 3*\|\mathbf{h}_{\mathrm{dl}}\|\}$
      elseif $\varrho < 0.25$
        $\Delta := \Delta/2$; *found* $:= (\Delta \le \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2))$
end


Q+A