



Lecture 5

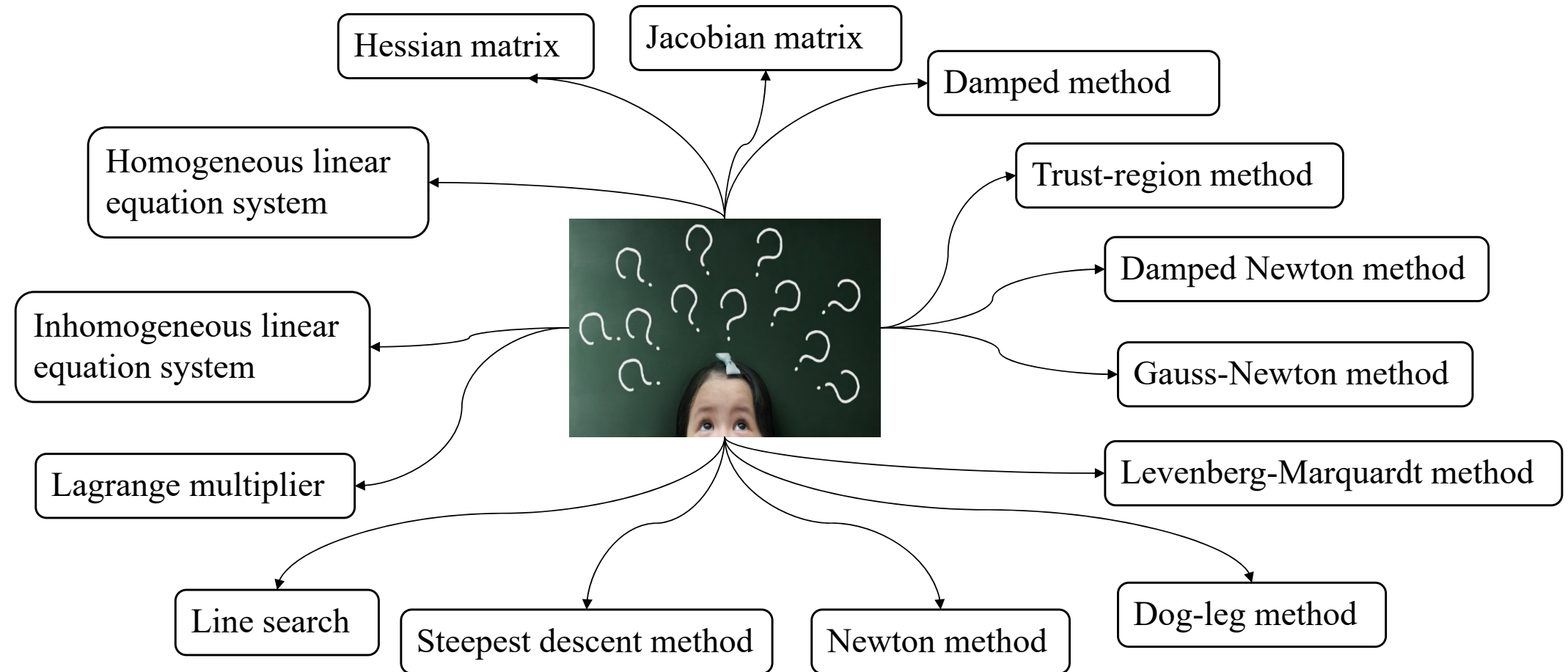
Math Prerequisite II: Nonlinear Least-squares

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



Why is least squares an important problem?

In engineering fields, some mathematical terminologies are often met





Outline

- 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

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

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,

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

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

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



Basic Concepts

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

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

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

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

Definition 2: Stationary point

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

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

- 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 $\mathbf{x}_1, \mathbf{x}_2, \dots$, 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} ,

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

Definition 3: Descent direction

\mathbf{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

$k := 0; \mathbf{x} := \mathbf{x}_0; found := \mathbf{false}$ {Starting point}

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

$\mathbf{h}_d := \text{search_direction}(\mathbf{x})$ {From \mathbf{x} and downhill}

if (no such \mathbf{h} exists)

$found := \mathbf{true}$ { \mathbf{x} is stationary}

else

$\alpha := \text{step_length}(\mathbf{x}, \mathbf{h}_d)$ {from \mathbf{x} in direction \mathbf{h}_d }

$\mathbf{x} := \mathbf{x} + \alpha \mathbf{h}_d; k := k + 1$ {next iterate}

end



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

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

$$\begin{aligned}\lim_{\alpha \rightarrow 0} \frac{F(\mathbf{x}) - F(\mathbf{x} + \alpha \mathbf{h})}{\alpha \|\mathbf{h}\|} &= \lim_{\alpha \rightarrow 0} \frac{F(\mathbf{x}) - [F(\mathbf{x}) + \alpha \mathbf{h}^T \mathbf{F}'(\mathbf{x})]}{\alpha \|\mathbf{h}\|} = -\frac{\mathbf{h}^T \mathbf{F}'(\mathbf{x})}{\|\mathbf{h}\|} \\ &= -\frac{\|\mathbf{h}\| \|\mathbf{F}'(\mathbf{x})\| \cos \theta}{\|\mathbf{h}\|} = -\|\mathbf{F}'(\mathbf{x})\| \cos \theta\end{aligned}$$

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.,

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

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,

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

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



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

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

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

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

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

else

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

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



2-phase methods: General Algorithm Framework

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

begin

$k := 0; \mathbf{x} := \mathbf{x}_0; found := \mathbf{false}$ {Starting point}

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

$\mathbf{h}_d := \text{search_direction}(\mathbf{x})$ {From \mathbf{x} and downhill}

if (no such \mathbf{h} exists)

$found := \mathbf{true}$ { \mathbf{x} is stationary}

else

$\alpha := \text{step_length}(\mathbf{x}, \mathbf{h}_d)$ {from \mathbf{x} in direction \mathbf{h}_d }

$\mathbf{x} := \mathbf{x} + \alpha \mathbf{h}_d; k := k + 1$ {next iterate}

end



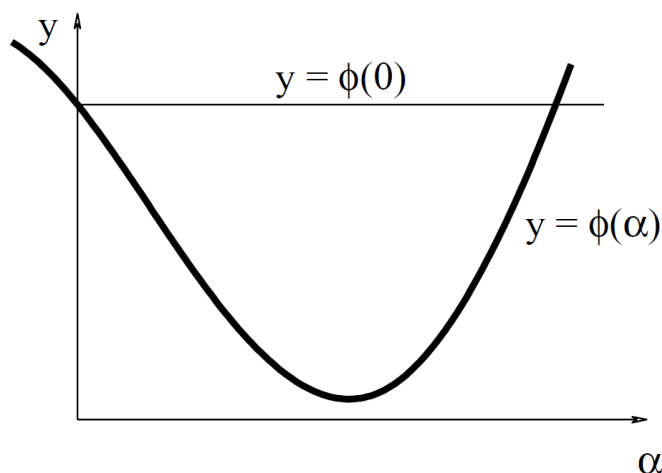
2-phase methods: Line search to find the step length

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 \mathbf{x} , and determine the step as

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



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

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 \mathbf{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 $\rho < 0.25$

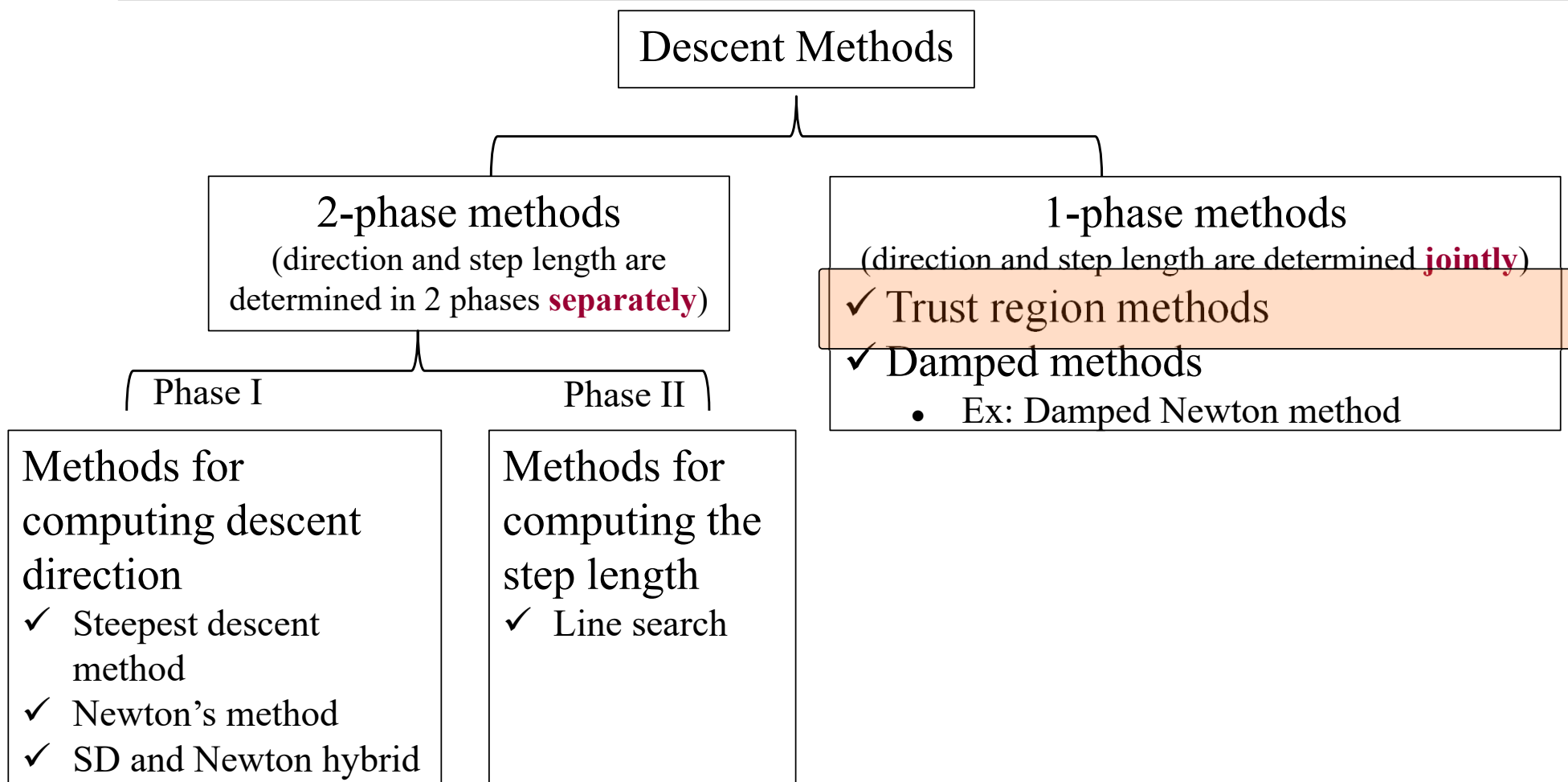
$\Delta := \Delta / 2$

elseif $\rho > 0.75$

$\Delta := \max \{ \Delta, 3 \times \|\mathbf{h}\| \}$



Descent Methods





1-phase methods: damped method

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

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

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

$$\begin{aligned}\phi'_\mu(\mathbf{h}) &= \frac{d\left(L(\mathbf{h}) + \frac{1}{2}\mu\mathbf{h}^T\mathbf{h}\right)}{d\mathbf{h}} = \frac{d\left(F(\mathbf{x}) + \mathbf{h}^T\mathbf{c} + \frac{1}{2}\mathbf{h}^T\mathbf{B}\mathbf{h} + \frac{1}{2}\mu\mathbf{h}^T\mathbf{h}\right)}{d\mathbf{h}} \\ &= \mathbf{c} + \frac{1}{2}(\mathbf{B} + \mathbf{B}^T)\mathbf{h} + \mu\mathbf{h} = \mathbf{c} + \mathbf{B}\mathbf{h} + \mu\mathbf{h} = 0\end{aligned}$$

➡ $\mathbf{h}_{dm} = -(\mathbf{B} + \mu\mathbf{I})^{-1}\mathbf{c} \quad (\text{Eq. 3})$



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 \mathbf{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

```
if  $\rho < 0.25$   
     $\mu := \mu \times 2$   
elseif  $\rho > 0.75$   
     $\mu := \mu / 3$   
(Marquart 1963)
```

Algo#5

The 2nd updating strategy for μ

```
 $v = 2$   
if  $\rho > 0$   
     $\mu := \mu \times \max \left\{ \frac{1}{3}, 1 - (2\rho - 1)^3 \right\}; v := 2$   
else  
     $\mu := \mu \times v; v := 2 \times v$   
(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,

$$\mathbf{h}_{dn} = -(\mathbf{F}''(\mathbf{x}) + \mu \mathbf{I})^{-1} \mathbf{F}'(\mathbf{x}) \quad \text{the so-called damped Newton step}$$

If μ is very large,

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

If μ is very small,

$$\mathbf{h}_{dn} \simeq -[\mathbf{F}''(\mathbf{x})]^{-1} \mathbf{F}'(\mathbf{x}), \text{ a step close to the Newton step}$$

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



Outline

- 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 $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $m \geq n$

We want to find,

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

where,

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

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

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

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



Basic Concepts

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



$$\begin{aligned} \frac{\partial F(\mathbf{x})}{\partial x_j} &= \frac{1}{2} \frac{\partial [f_1^2(\mathbf{x}) + f_2^2(\mathbf{x}) + \dots + f_m^2(\mathbf{x})]}{\partial x_j} \\ &= f_1(\mathbf{x}) \frac{\partial f_1(\mathbf{x})}{\partial x_j} + f_2(\mathbf{x}) \frac{\partial f_2(\mathbf{x})}{\partial x_j} + \dots + f_m(\mathbf{x}) \frac{\partial f_m(\mathbf{x})}{\partial x_j} \\ &= \sum_{i=1}^m \left[f_i(\mathbf{x}) \frac{\partial f_i(\mathbf{x})}{\partial x_j} \right] \end{aligned}$$



Basic Concepts

$$\begin{aligned} \mathbf{F}'(\mathbf{x}) &= \begin{bmatrix} \frac{\partial F(\mathbf{x})}{\partial x_1} \\ \frac{\partial F(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial F(\mathbf{x})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{x}) \frac{\partial f_1}{\partial x_1} + f_2(\mathbf{x}) \frac{\partial f_2}{\partial x_1} + \dots + f_m(\mathbf{x}) \frac{\partial f_m}{\partial x_1} \\ f_1(\mathbf{x}) \frac{\partial f_1}{\partial x_2} + f_2(\mathbf{x}) \frac{\partial f_2}{\partial x_2} + \dots + f_m(\mathbf{x}) \frac{\partial f_m}{\partial x_2} \\ \vdots \\ f_1(\mathbf{x}) \frac{\partial f_1}{\partial x_n} + f_2(\mathbf{x}) \frac{\partial f_2}{\partial x_n} + \dots + f_m(\mathbf{x}) \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{n \times m} \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{bmatrix} \\ &= (\mathbf{J}(\mathbf{x}))^T \mathbf{f}(\mathbf{x}) \end{aligned} \quad (\text{Eq. 5})$$



Basic Concepts

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



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



$$\mathbf{F}''(\mathbf{x}) = \underbrace{(\mathbf{J}(\mathbf{x}))^T}_{n \times m} \underbrace{\mathbf{J}(\mathbf{x})}_{m \times n} + \sum_{i=1}^m \underbrace{f_i(\mathbf{x})}_{1 \times 1} \underbrace{\mathbf{f}_i''(\mathbf{x})}_{n \times n} \quad (\text{addition of a stack of matrices})$$



Outline

- 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 \mathbf{f} (a linear model of \mathbf{f}) in the neighborhood of \mathbf{x} (refer to Eq. 4),

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

We suppose \mathbf{J} has full column rank


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

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,

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

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

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

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

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



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**.

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

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

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

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

$\mathbf{J}^T \mathbf{J}$ is positive definite



Outline


- 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 \mathbf{f} (a linear model of \mathbf{f}) in the neighborhood of \mathbf{x} , $\mathbf{f}(\mathbf{x} + \mathbf{h}) \simeq \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x})\mathbf{h}$ **We don't require \mathbf{J} has full column rank**


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

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

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



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

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

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:

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

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



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

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



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



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



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



Levenberg-Marquardt Method

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

L-M's step:

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

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

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

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 $(\mathbf{J}(\mathbf{x}_0))^T \mathbf{J}(\mathbf{x}_0)$ 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

$$\|\mathbf{F}'(\mathbf{x})\|_{\infty} \leq \varepsilon_1$$

as the first stopping criterion

- If for the current iteration, the change of \mathbf{x} is too small,

$$\|\mathbf{x}_{new} - \mathbf{x}\|_2 \leq \varepsilon_2 (\|\mathbf{x}\|_2 + \varepsilon_2)$$

- Finally, we need a safeguard against an infinite loop,

$$k \geq k_{\max}$$

where k is the current iteration index



Levenberg-Marquardt Method

Algo#6: L-M Method

begin

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

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

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

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

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

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

$found := \mathbf{true}$

else

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

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

if $\varrho > 0$

{step acceptable}

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

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

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

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

else

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

end

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



Outline

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

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

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

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



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

This function of α is minimal for,

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



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 $\|\mathbf{h}_{gn}\| \leq \Delta$

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

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

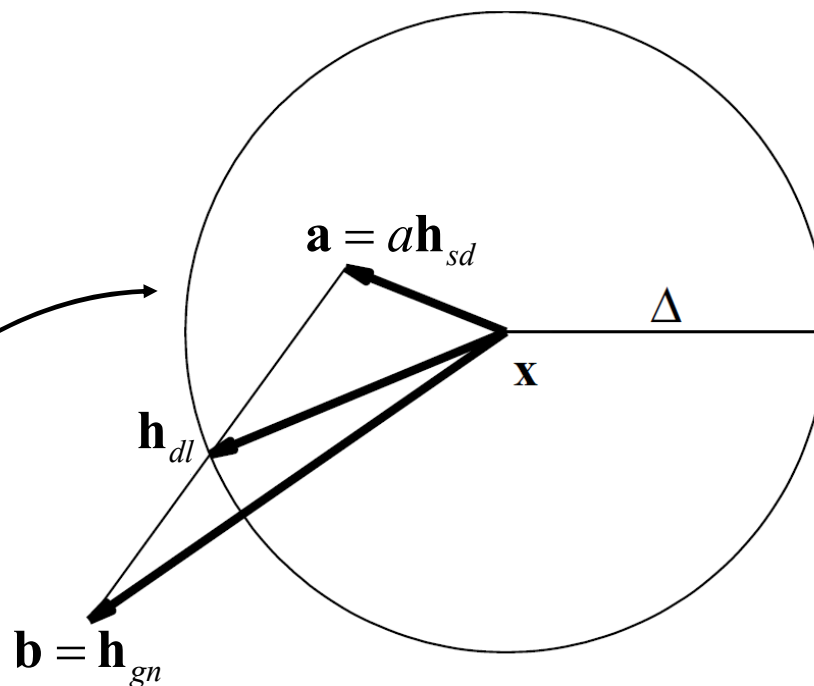
$$\mathbf{h}_{dl} := \frac{\Delta}{\|\mathbf{h}_{sd}\|} \mathbf{h}_{sd}$$

else

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

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

the last case



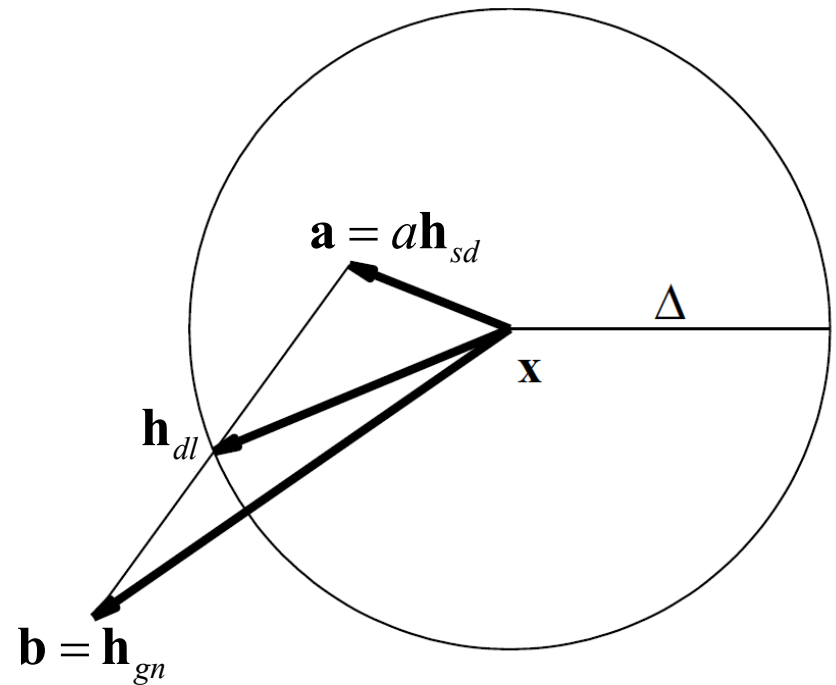


Powell's Dog Leg Method

The name *Dog Leg* is taken from golf: The fairway at a “dog leg hole” has a shape as the line from \mathbf{x} (the tee point) via the end point of \mathbf{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

$k := 0; \quad \mathbf{x} := \mathbf{x}_0; \quad \Delta := \Delta_0; \quad \mathbf{g} := \mathbf{J}(\mathbf{x})^\top \mathbf{f}(\mathbf{x})$

$found := (\|\mathbf{f}(\mathbf{x})\|_\infty \leq \varepsilon_3) \text{ or } (\|\mathbf{g}\|_\infty \leq \varepsilon_1)$

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

$k := k+1; \quad$ Compute α by (Eq. 6)

$\mathbf{h}_{sd} := -\alpha \mathbf{g}; \quad$ Solve $\mathbf{J}(\mathbf{x})\mathbf{h}_{gn} \simeq -\mathbf{f}(\mathbf{x})$

Compute \mathbf{h}_{dl} by (Algo# 6)

if $\|\mathbf{h}_{dl}\| \leq \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2)$

$found := \text{true}$

else

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

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

if $\rho > 0$

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

$found := (\|\mathbf{f}(\mathbf{x})\|_\infty \leq \varepsilon_3) \text{ or } (\|\mathbf{g}\|_\infty \leq \varepsilon_1)$

if $\rho > 0.75$

$\Delta := \max\{\Delta, 3*\|\mathbf{h}_{dl}\|\}$

elseif $\rho < 0.25$

$\Delta := \Delta/2; \quad found := (\Delta \leq \varepsilon_2(\|\mathbf{x}\| + \varepsilon_2))$

end

