Lecture 5
Math Prerequisite II: Nonlinear Least-squares

Lin ZHANG, PhD
School of Software Engineering
Tongji University
Spring, 2020
Why is least squares an important problem?

In engineering fields, some mathematical terminologies are often met:

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

• 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,

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

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

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

It is easy to verify that,

$$F''(x) = \frac{dF'(x)}{dx^T}$$
Basic Concepts

**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**.
Basic Concepts

**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

• 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 \(x_0\) the method produces a series of vectors \(x_1, x_2, \ldots\), 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
Descent Methods

Consider the variation of the $F$-value along the half line staring at $x$ and with direction $h$,

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

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

**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 \textit{separately})

1-phase methods
(direction and step length are determined \textit{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
Algo#1: 2-phase Descent Method (a general framework)

begin
\[ k := 0; \; x := x_0; \; found := \text{false} \]

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

\[ h_d := \text{search\_direction}(x) \]

if (no such \( h \) exists)

\[ found := \text{true} \]

else

\[ \alpha := \text{step\_length}(x, h_d) \]

\[ x := x + \alpha h_d; \; k := k + 1 \]

end
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 **steeppest 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 $x^*$ is a stationary point, i.e.,

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

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

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

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

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

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
\[
x := x + h_n
\]
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

```plaintext
if F''(x) is positive definite
    h_d := h_n
else
    h_d := h_{sd}
end if
x := x + \alpha h_d
```
Algo#1: 2-phase Descent Method (a general framework)

begin
\[ k := 0; \quad x := x_0; \quad found := \text{false} \quad \text{\{Starting point\}} \]
while (not found) and (\(k < k_{\text{max}}\))
\[ h_d := \text{search\_direction}(x) \quad \text{\{From x and downhill\}} \]
if (no such \(h\) exists)
\[ found := \text{true} \quad \text{\{x is stationary\}} \]
else
\[ \alpha := \text{step\_length}(x, h_d) \quad \text{\{from x in direction \(h_d\)\}} \]
\[ x := x + \alpha h_d; \quad k := k + 1 \quad \text{\{next iterate\}} \]
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 \( h \),

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

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
• 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

1-phase methods
(direction and step length are determined **jointly**)
- Trust region methods
- Damped methods
  - Ex: Damped Newton method

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

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

Methods for computing the step length
- Line search
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 $x$,

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

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

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

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

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 \( \mathbf{h} \) 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 \( \mathbf{h}_{tr} \) 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 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 $\rho > 0.75$
\[ \Delta := \max \{ \Delta, 3\|h\| \} \]
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,

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

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

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

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

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

**Algo#4**

The 1\textsuperscript{st} updating strategy for $\mu$

\[
\begin{align*}
\text{if } \rho &< 0.25 \\
\mu &:= \mu \times 2 \\
\text{elseif } \rho &> 0.75 \\
\mu &:= \mu / 3
\end{align*}
\]

(Marquardt 1963)

**Algo#5**

The 2\textsuperscript{nd} updating strategy for $\mu$

\[
\begin{align*}
\text{if } \rho &> 0 \\
\mu &:= \mu \times \max \left\{ \frac{1}{3}, 1 - (2\rho - 1)^3 \right\}; v := 2 \\
\text{else} \\
\mu &:= \mu \times v; v := 2 \times v
\end{align*}
\]

(Nielsen 1999)
1-phase methods: damped method

Ex: Damped Newton method

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

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

(Eq. 3) takes the form,

\[ h_{dn} = -\left(F''(x) + \mu I\right)^{-1} F'(x) \quad \text{the so-called damped Newton step} \]

If \( \mu \) is very large,

\[ h_{dn} \approx -\frac{1}{\mu} F'(x) , \text{ a short step in a direction close to the deepest descent direction} \]

If \( \mu \) is very small,

\[ h_{dn} \approx -\left[F''(x)\right]^{-1} F'(x) , \text{ a step close to the Newton step} \]

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 \( f : \mathbb{R}^n \rightarrow \mathbb{R}^m, m \geq n \)

We want to find,
\[
\mathbf{x}^* = \text{arg} \min_x \left\{ F(\mathbf{x}) \right\}
\]

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, but there are more efficient methods for dealing with this special kind of nonlinear optimization problems
Basic Concepts

Taylor expansion for \( f(x) \),

\[
\begin{bmatrix}
    f_1(x+h) \\
    f_2(x+h) \\
    \vdots \\
    f_m(x+h)
\end{bmatrix} = 
\begin{bmatrix}
    f_1(x) + (\nabla f_1(x))^T h + O(h^2) \\
    f_2(x) + (\nabla f_2(x))^T h + O(h^2) \\
    \vdots \\
    f_m(x) + (\nabla f_m(x))^T h + O(h^2)
\end{bmatrix} = 
\begin{bmatrix}
    f_1(x) \\
    f_2(x) \\
    \vdots \\
    f_m(x)
\end{bmatrix} + 
\begin{bmatrix}
    (\nabla f_1(x))^T \\
    (\nabla f_2(x))^T \\
    \vdots \\
    (\nabla f_m(x))^T
\end{bmatrix} h + O(h^2)
\]

\( = f(x) + J(x) h + O(h^2) \) \hspace{1cm} (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

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

\[
= (J(x))^Tf(x)
\]

(Eq. 5)
Basic Concepts

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

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

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

\[ n \times m \quad m \times n \quad 1 \times 1 \quad n \times n \]
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
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),

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

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

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

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

\( h_{gn} \) is the solution to,

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

\[
\Rightarrow \quad h_{gn} = - (J^T J)^{-1} J^T f
\]

**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 found by line search
Gauss-Newton Method

• Some notes about Gauss-Newton methods
  – The **classical Gauss-Newton method** uses $\alpha = 1$ in all steps
  – 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 $\iff$ $\mathbf{J}$’s columns are linearly unrelated

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

$\mathbf{J}^T\mathbf{J}$ 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 $f$ (a linear model of $f$) in the neighborhood of $x$, \[ f(x + h) = f(x) + J(x)h \]

We don’t require $J$ has full column rank

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

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

\[
h_{lm} = \arg \min_h L(h) + \frac{1}{2} \mu h^T h , \text{ where } \mu > 0 \text{ is the damped coefficient}
\]

$h_{lm}$ is the solution to,

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

positive definite
Levenberg-Marquardt Method

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 $\{\lambda_i \geq 0, i = 1, ..., N\}$

$A v_i = \lambda_i v_i$

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

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 \( \alpha = 1 \)):

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

That’s why we say L-M is a damped Gauss-Newton method
Levenberg-Marquardt Method

• Updating strategy of $\mu$
  – $\mu$ influences both the direction and the size of the step, and this leads L-M \textbf{without} a specific line search
  – The initial $\mu$–value is related to the elements in $(J(x_0))^T J(x_0)$ by letting,
    \[ \mu_0 = \tau \cdot \max_i \left\{ \left( J^T J \right)_{ii}^{(0)} \right\} \]
  – During iteration, $\mu$ can be updated by \textbf{Algo#4} or \textbf{Algo#5}
• Stopping criteria
  – For a minimizer $x^*$, ideally we will have $F'(x^*)=0$

  So, we can use
  \[ \| F'(x) \|_\infty \leq \varepsilon_1 \]
  as the first stopping criterion
  – If for the current iteration, the change of $x$ is too small,
  \[ \| x_{new} - x \|_2 \leq \varepsilon_2 \left( \| x \|_2 + \varepsilon_2 \right) \]
  – 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
    \( k := 0; \quad \nu := 2; \quad x := x_0 \)
    \( A := J(x) \top J(x); \quad g := J(x) \top f(x) \)
    \( \text{found} := (\|g\|_\infty \leq \varepsilon_1); \quad \mu := \tau \times \max\{a_{ii}\} \)
    while (not found) and (\( k < k_{\text{max}} \))
        \( k := k + 1; \quad \text{Solve} \ (A + \mu I)h_{ln} = -g \)
        if \( \|h_{ln}\| \leq \varepsilon_2(\|x\| + \varepsilon_2) \)
            \( \text{found} := \text{true} \)
        else
            \( x_{\text{new}} := x + h_{ln} \)
            \( \rho := (F(x) - F(x_{\text{new}}))/(L(0) - L(h_{ln})) \)
            if \( \rho > 0 \) \{step acceptable\}
                \( x := x_{\text{new}} \)
                \( A := J(x) \top J(x); \quad g := J(x) \top f(x) \)
                \( \text{found} := (\|g\|_\infty \leq \varepsilon_1) \)
                \( \mu := \mu \times \max\{\frac{1}{3}, 1 - (2\rho - 1)^3\}; \quad \nu := 2 \)
            else
                \( \mu := \mu \times \nu; \quad \nu := 2 \times \nu \)

end
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

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

Powell is a keen golfer!
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,

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

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

This function of \( \alpha \) is minimal for,

\[
\alpha = \frac{-h_{sd}^T J^T f}{h_{sd}^T J^T J h_{sd}} = \frac{F'(x)^T F'(x)}{h_{sd}^T J^T J h_{sd}} = \frac{\|F'(x)\|^2}{h_{sd}^T J^T J h_{sd}}
\] (Eq. 6)
Powell’s Dog Leg Method

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

\[
\mathbf{a} = \alpha \mathbf{h}_{sd}, \quad \mathbf{b} = \mathbf{h}_{gn}
\]

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

**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

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

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