Chapter 09
Nonlinear Least-squares

Lin ZHANG
School of Computer Science and Technology
Tongji University

Why is nonlinear least-squares an important problem?

Many engineering problems can be formulated as nonlinear least-squares problems

PnP problem: $$\xi^* = \arg\min_{\xi} \frac{1}{2} \sum_{i=1}^{m} \left\| \mathbf{u}_i - \frac{1}{s_i} \mathbf{K} \left(\exp\left(\xi^\wedge\right) \begin{pmatrix} \mathbf{p}_i \\ 1 \end{pmatrix} \right)_{1:3} \right\|_2^2$$

ICP problem: $$\xi^* = \arg\min_{\xi} \frac{1}{2} \sum_{i=1}^{n} \left\| \mathbf{p}_i - \left(\exp\left(\xi^\wedge\right) \begin{pmatrix} \mathbf{p}_i' \\ 1 \end{pmatrix} \right)_{1:3} \right\|_2^2$$

Direct method: $$\xi^* = \arg\min_{\xi} \frac{1}{2} \sum_{i=1}^{m} \left\| \mathbf{I}_2 \left(\frac{1}{s_i} \mathbf{K} \left(\exp\left(\xi^\wedge\right) \begin{pmatrix} \mathbf{p}_i \\ 1 \end{pmatrix} \right)_{1:3} \right) - \mathbf{I}_1 \left(\mathbf{u}_i^1 \right) \right\|_2^2$$

Camera calibration: $$\theta^* = \arg\min_{\theta} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} \left\| \mathbf{K} \cdot \mathcal{D} \left\{ \frac{1}{z_{cij}} \left[\mathcal{R}\left(\mathbf{d}_i\right), \mathbf{t}_i \right] \mathbf{p}_j \right\} - \mathbf{u}_{ij} \right\|_2^2$$

All these problems actually can be solved by a unified algorithm

Outline

- Pre-requisites
- Unconstrained optimization
  - Problem definition
  - Damped method
- Non-linear Least Squares
  - Problem definition
  - Gauss-Newton method
  - Levenberg-Marquardt method

Pre-requisites

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

Pre-requisites

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

$$f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + \mathbf{h}^T \nabla f(\mathbf{x}) + \frac{1}{2}\mathbf{h}^T \nabla^2 f(\mathbf{x})\mathbf{h} + O\left(\|\mathbf{h}\|^2\right)$$

where $\nabla f(\mathbf{x})$ is the gradient and $\nabla^2 f(\mathbf{x})$ is the Hessian,

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

Pre-requisites

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

$$f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + \mathbf{h}^T \nabla f(\mathbf{x}) + \frac{1}{2}\mathbf{h}^T \nabla^2 f(\mathbf{x})\mathbf{h} + O\left(\|\mathbf{h}\|^2\right)$$

where $\nabla f(\mathbf{x})$ is the gradient and $\nabla^2 f(\mathbf{x})$ is the Hessian,

It is easy to verify that,

$$\nabla^2 f(\mathbf{x}) = \frac{d\nabla f(\mathbf{x})}{d\mathbf{x}^T}$$

Pre-requisites

Theorem 1: Necessary condition for a local minimizer

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

$$\nabla f\left(\mathbf{x}^*\right)=\mathbf{0}$$

Definition 2: Stationary point

If $\nabla f\left(\mathbf{x}_s\right)=\mathbf{0}$,

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**

Pre-requisites

Theorem 2: Sufficient condition for a local minimizer

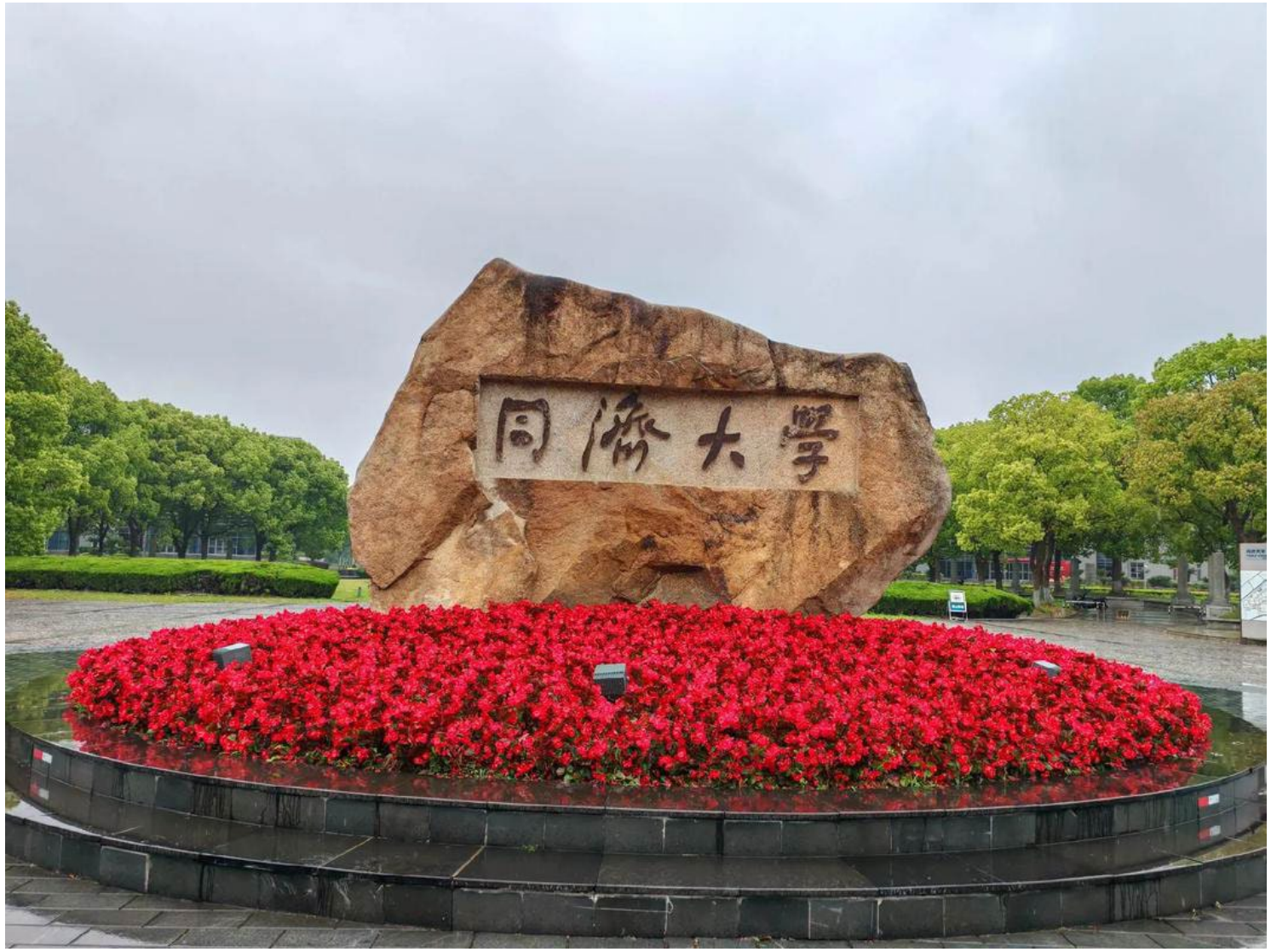
Assume that $\mathbf{x}_s$ is a stationary point and that $\nabla^2 f(\mathbf{x}_s)$ is positive definite, then $\mathbf{x}_s$ is a local minimizer

If $\nabla^2 f(\mathbf{x}_s)$ is negative definite, then $\mathbf{x}_s$ is a local maximizer. If $\nabla^2 f(\mathbf{x}_s)$ is indefinite (i.e. it has both positive and negative eigenvalues), then $\mathbf{x}_s$ is a saddle point

Outline

- Pre-requisites
- Unconstrained optimization
  - Problem definition
  - Damped method
- Non-linear Least Squares
  - Problem definition
  - Gauss-Newton method
  - Levenberg-Marquardt method

Unconstrained optimization—problem definition



Suppose you are visiting Tongji University's Jiading Campus for the first time, and I assign you a task: to find the lowest point on this campus. What would you do?

Unconstrained optimization—problem definition

Suppose $f(\mathbf{x}):|\mathbb{R}^n \to \mathbb{R}$ is a differentiable function. In the general case (e.g., $f(\mathbf{x})$ is very complex, or we don't know its convexity), it is nearly impossible to get its global minimizer

↓

A local minimizer (usually it is good enough)

↓

The local minimizer you find for $f(\mathbf{x})$ depends on the initial position $\mathbf{x}_0$ and the concrete optimization method you use

↓

The problem to be solved in this lecture:
Starting from an initial point $\mathbf{x}_0$, finding the local minimizer of the function $f(\mathbf{x})$ through iterative optimization

Unconstrained optimization—problem definition

- In general, the optimization process is iterative: starting from the initial point $\mathbf{x}_0$, the algorithm generates a new iteration point $\mathbf{x}_1,\mathbf{x}_2,\ldots$ after each iteration; we hope that this process can be completed within a finite number of steps and eventually converge to a minimizer $\mathbf{x}^*$ of the function $f(\mathbf{x})$
- During this process, the algorithm needs to ensure that iterations continuously reduce the function value,

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

- At each step of an iterative algorithm, the essence is to determine the update vector: for the current iteration point $\mathbf{x}_k$, we need to identify the update vector $\mathbf{h}$, and then obtain the next iteration point $\mathbf{x}_{k+1}= \mathbf{x}_k+\mathbf{h}$
- The differences among various iterative optimization algorithms lie in how they calculate the update vector in each iteration

We will learn an intuitive and commonly used iterative optimization framework: damped method

Outline

- Pre-requisites
- Unconstrained optimization
  - Problem definition
  - Damped method
- Non-linear Least Squares
  - Problem definition
  - Gauss-Newton method
  - Levenberg-Marquardt method

Damped method

Our problem is to determine the update vector $\mathbf{h}_{\mathrm{dm}}$ of the function $f(\mathbf{x}):|\mathbb{R}^n \to \mathbb{R}$ at the current point $\mathbf{x}_k$ so as to obtain the next iteration point $\mathbf{x}_k+\mathbf{h}_{\mathrm{dm}}$

Core idea: we create a surrogate function $l(\mathbf{h})$ for $f(\mathbf{x}_k+\mathbf{h})$ based on the local information of $f(\mathbf{x}_k)$. $l(\mathbf{h})$ "looks very similar" to $f(\mathbf{x}_k+\mathbf{h})$ locally (when $||\mathbf{h}||$ is small); in addition, $l(\mathbf{h})$ should be extremely simple—simple enough that we can easily get its global minimizer $\mathbf{h}_{\mathrm{dm}}$. Then, we trust the suggestion of this surrogate $l(\mathbf{h})$ and update the next iteration point of $f(\mathbf{x})$ to $\mathbf{x}_{k+1}=\mathbf{x}_k+\mathbf{h}_{\mathrm{dm}}$

Can you imagine what is the form of $l(\mathbf{h})$?

A quadratic function is a good choice and we require $l(\mathbf{0})=f(\mathbf{x}_k)$

Damped method

Our problem is to determine the update vector $\mathbf{h}_{\mathrm{dm}}$ of the function $f(\mathbf{x}):|\mathbb{R}^n \to \mathbb{R}$ at the current point $\mathbf{x}_k$ so as to obtain the next iteration point $\mathbf{x}_k+\mathbf{h}_{\mathrm{dm}}$

The local surrogate function $l(\mathbf{h})$,

$$l(\mathbf{h}) = f(\mathbf{x}_k) + \mathbf{h}^T\mathbf{c} + \frac{1}{2}\mathbf{h}^T\mathbf{B}\mathbf{h} \text{ , where } \mathbf{c} \in \mathbb{R}^n \text{ and } \mathbf{B} \in \mathbb{R}^{n\times n} \text{ is symmetric } \textbf{(Eq. 1)}$$

$\mathbf{c}$ and $\mathbf{B}$ are constructed based on local information of f at $\mathbf{x}_k$ (will be discussed later)

Then,

$$\mathbf{h}_{\mathrm{dm}} = \arg\min_{\mathbf{h}} l(\mathbf{h})$$

Besides, large $\|\mathbf{h}\|$ should be penalized, since $l(\mathbf{h})$ can approximate $f(\mathbf{x}_k+\mathbf{h})$ well only when $\mathbf{h}$ is relatively small

Damped method

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

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

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

$$l(\mathbf{h}) + \frac{1}{2}\mu \mathbf{h}^T \mathbf{h}$$

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

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

Damped method

$$\frac{d\left(l(\mathbf{h})+\frac{1}{2}\mu\mathbf{h}^T\mathbf{h}\right)}{d\mathbf{h}}=\frac{d\left(f(\mathbf{x}_k)+\mathbf{h}^T\mathbf{c}+\frac{1}{2}\mathbf{h}^T\mathbf{B}\mathbf{h}+\frac{1}{2}\mu\mathbf{h}^T\mathbf{h}\right)}{d\mathbf{h}}$$

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

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

In a concrete algorithm, $\mathbf{B}+\mu\mathbf{I}$ is usually constructed to be positive definite When $\mathbf{B}+\mu\mathbf{I}$ is positive definite, $l(\mathbf{h})+1/2\mu\mathbf{h}^T\mathbf{h}$ actually is a (strictly) convex function, so its stationary point $-(\mathbf{B}+\mu\mathbf{I})^{-1}\mathbf{c}$ is its global minimizer

Damped method—Updating

- The algorithm will accept $\mathbf{h}_{dm}$ and proceed to the next iteration point $\mathbf{x}_k+\mathbf{h}_{dm}$ only if $f(\mathbf{x}_k+\mathbf{h}_{dm})<f(\mathbf{x}_k)$; otherwise, no update of the iteration point will be performed
- Regardless of whether the $\mathbf{h}_{dm}$ calculated in the current step is accepted or not, it is necessary to adjust μ for the calculation of the update vector in the next step

Algo#1 Basic steps using a damped method

compute $\mathbf{h}_{dm}$ by Eq. 3
if $f(\mathbf{x}_k+\mathbf{h}_{dm})<f(\mathbf{x}_k)$
  $\mathbf{x}_{k+1}:=\mathbf{x}_k+\mathbf{h}_{dm}$
update μ

the core problem

Damped method—update μ

- If the current update vector obtained is *not good enough*, we need to increase μ to increase the penalty to large steps

The quality of the current update vector can be evaluated by the **gain ratio**,

Definition 3: Gain ratio

$$\rho = \frac{f(\mathbf{x}_k) - f(\mathbf{x}_k + \mathbf{h}_{\mathrm{dm}})}{l(\mathbf{0}) - l(\mathbf{h}_{\mathrm{dm}})}$$

the actual decrease

the predicted decrease

This part is constructed to be positive. Why?

Damped method—update μ

- 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 $f(\mathbf{x}_k+\mathbf{h})$, and μ may be reduced

Algo#2

The 1st updating strategy for μ

if $\rho < 0.25$

$\mu := \mu \times 2$

elseif $\rho > 0.75$

$\mu := \mu / 3$

(Marquart 1963)

Algo#3

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)

Damped method

Ex: Damped Newton method

$$l(\mathbf{h}) = f(\mathbf{x}_k) + \mathbf{h}^T\mathbf{c} + \frac{1}{2}\mathbf{h}^T\mathbf{B}\mathbf{h}$$

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

if $\mathbf{c} = \nabla f(\mathbf{x}_k)$ and $\mathbf{B} = \nabla^2 f(\mathbf{x}_k)$

(Eq. 3) takes the following concrete form,

$$\mathbf{h}_{\text{dn}} = -\left(\nabla^2 f(\mathbf{x}) + \mu\mathbf{I}\right)^{-1} \nabla f(\mathbf{x})$$

the so-called damped Newton step

Outline

- Pre-requisites
- Unconstrained optimization
  - Problem definition
  - Damped method
- Non-linear Least Squares
  - Problem definition
  - Gauss-Newton method
  - Levenberg-Marquardt method

Problem definition

- Formulation of the non-linear least squares problem

Given a vector function $\mathbf{f}(\mathbf{x}) = \left(f_1(\mathbf{x}), f_2(\mathbf{x}), .., f_m(\mathbf{x})\right)^T$, $\mathbf{x} \in \mathbb{R}^n$, $f_i(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, we want to find,

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

where,

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}\left(f_i(\mathbf{x})\right)^2 = \frac{1}{2}\left\|\mathbf{f}(\mathbf{x})\right\|_2^2 = \frac{1}{2}\mathbf{f}(\mathbf{x})^T\,\mathbf{f}(\mathbf{x})$$

- Non-linear least squares problems can be solved by the general damped method, which will have some specific forms for this special case

Problem definition

1st-order Taylor expansion for $\mathbf{f}(\mathbf{x}): \mathbb{R}^n \to \mathbb{R}^m$,

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

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

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

Problem definition

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

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

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

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

Problem definition

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

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

Outline

- Pre-requisites
- Unconstrained optimization
  - Problem definition
  - Damped method
- Non-linear Least Squares
  - Problem definition
  - Gauss-Newton method
  - Levenberg-Marquardt method

Gauss-Newton method

At the current iteration point $\mathbf{x}_k$, how to compute the update step vector $\mathbf{h}_{gn}$?

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 $\mathbf{x}_k$ (refer to Eq. 4),

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

We suppose **J** has full column rank

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

$$= f(\mathbf{x}_k)+\mathbf{h}^T\mathbf{J}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)+\frac{1}{2}\mathbf{h}^T\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)\mathbf{h}$$

Take it as the surrogate function $l(\mathbf{h})$ for $f(\mathbf{x}_k+\mathbf{h})$

$$l(\mathbf{h}) \equiv f(\mathbf{x}_k)+\mathbf{h}^T\underbrace{\mathbf{J}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)}_{\mathbf{c}}+\frac{1}{2}\mathbf{h}^T\underbrace{\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)}_{\mathbf{B}}\mathbf{h}$$

Compare it with Eq. (1) and Eq. (5)

$$\nabla f(\mathbf{x}_k)$$

Gauss-Newton method

The Gauss-Newton update step $\mathbf{h}_{gn}$ defined as the vector that minimizes $l(\mathbf{h})$,

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

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

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

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

It can be considered that the Gauss-Newton's updating step is obtained by the damped method with μ=0 (compare it with Eq. 3)

Gauss-Newton method

- Some notes about Gauss-Newton methods
  - For each iteration step, it requires that the Jacobian **J** has full column rank

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

Proof:

J has full column rank $\Leftrightarrow$ **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

- Pre-requisites
- Unconstrained optimization
  - Problem definition
  - Damped method
- Non-linear Least Squares
  - Problem definition
  - Gauss-Newton method
  - Levenberg-Marquardt method

Levenberg-Marquardt method

Goal: at the current iteration point $\mathbf{x}_k$, compute the update step vector

Gauss-Newton method

Local surrogate function $l(\mathbf{h})$ for $f(\mathbf{x}_k+\mathbf{h})$

$$l(\mathbf{h}) \equiv f(\mathbf{x}_k) + \mathbf{h}^T \mathbf{J}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k) + \frac{1}{2}\mathbf{h}^T \mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)\mathbf{h}$$

Update step is computed as,

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

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

We suppose $\mathbf{J}(\mathbf{x}_k)$ has full column rank

Levenberg-Marquardt method

Local surrogate function $l(\mathbf{h})$ for $f(\mathbf{x}_k+\mathbf{h})$

$$l(\mathbf{h}) \equiv f(\mathbf{x}_k) + \mathbf{h}^T \mathbf{J}^T(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k) + \frac{1}{2}\mathbf{h}^T \mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)\mathbf{h}$$

Update step is computed as,

$$\mathbf{h}_{\text{lm}} = \arg\min_{\mathbf{h}} \left\{ l(\mathbf{h}) + \frac{1}{2}\mu\mathbf{h}^T\mathbf{h} \right\}, \mu > 0$$

positive definite

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

We don't require $\mathbf{J}(\mathbf{x}_k)$ has full column rank

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 $\{\lambda_i \geq 0, i = 1, \ldots, n\}$

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

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

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:

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

Gauss-Newton's step:

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

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 $\left(\mathbf{J}\left(\mathbf{x_0}\right)\right)^T \mathbf{J}\left(\mathbf{x_0}\right)$ by letting,

$$\mu_0 = \tau \cdot \max_i \left\{ \left(\mathbf{J}^T \left(\mathbf{x}_0 \right) \mathbf{J} \left(\mathbf{x}_0 \right) \right)_{ii} \right\}, i = 1, \ldots, n$$

  - During iteration, μ can be updated by **Algo#2** or **Algo#3**

Levenberg-Marquardt method

- Stopping criteria
  - For a minimizer $\mathbf{x}^*$, ideally we will have $\nabla f\left(\mathbf{x}^*\right)=0$

    So, we can use

    $$\left\|\nabla f\left(\mathbf{x}\right)\right\|_\infty \le \varepsilon_1$$

    as the first stopping criterion
  - If for the current iteration at $\mathbf{x}$, the update step $\mathbf{h}_{\text{lm}}$ is too small,

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

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

    where k is the current iteration index

Levenberg-Marquardt method

Algo#4: L-M Method

begin

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

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

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

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

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

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

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

$found := \mathbf{true}$

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}

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

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

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

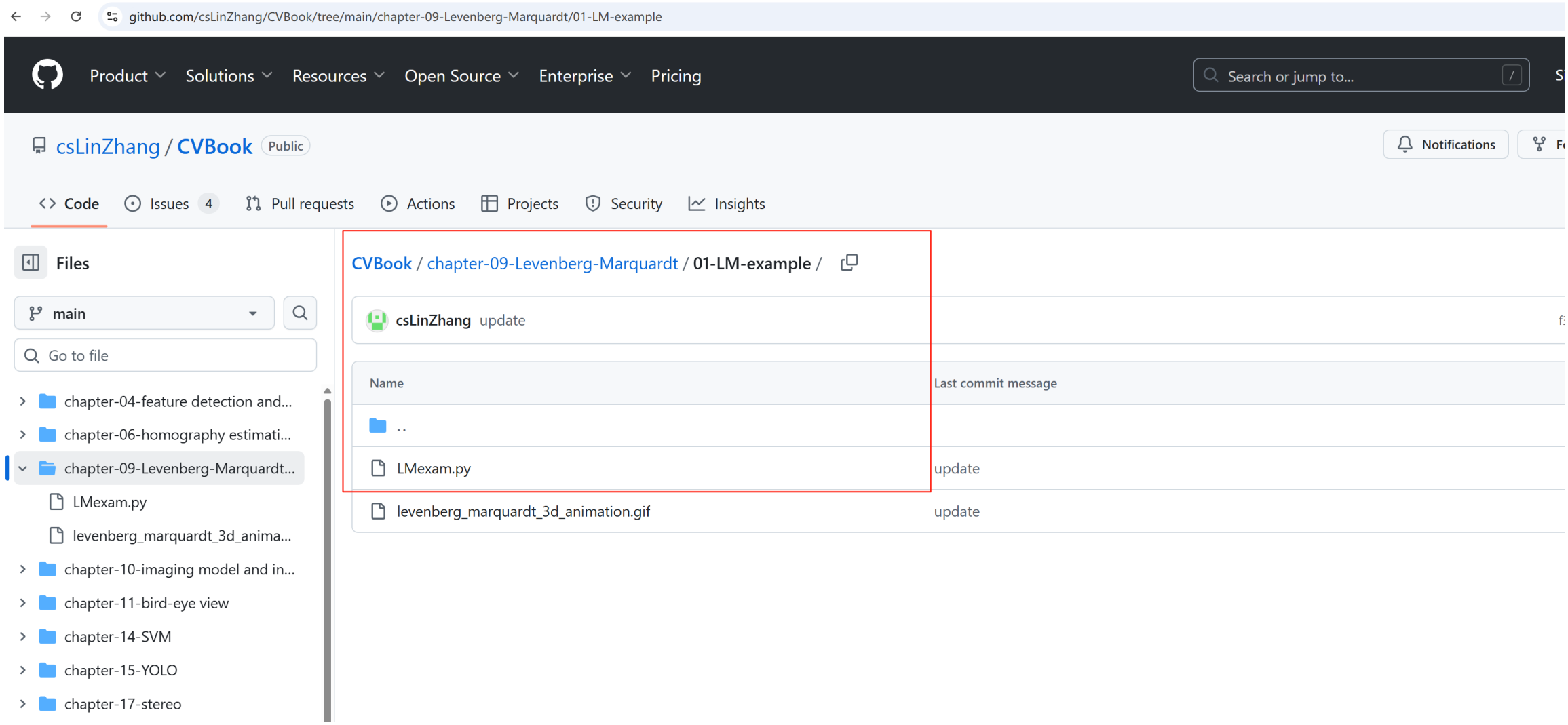
$\mu := \mu * \max\{\frac{1}{3}, 1 - (2\varrho - 1)^3\}; \quad \nu := 2$

else

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

end

Levenberg-Marquardt method—An example



Levenberg-Marquardt method—An example

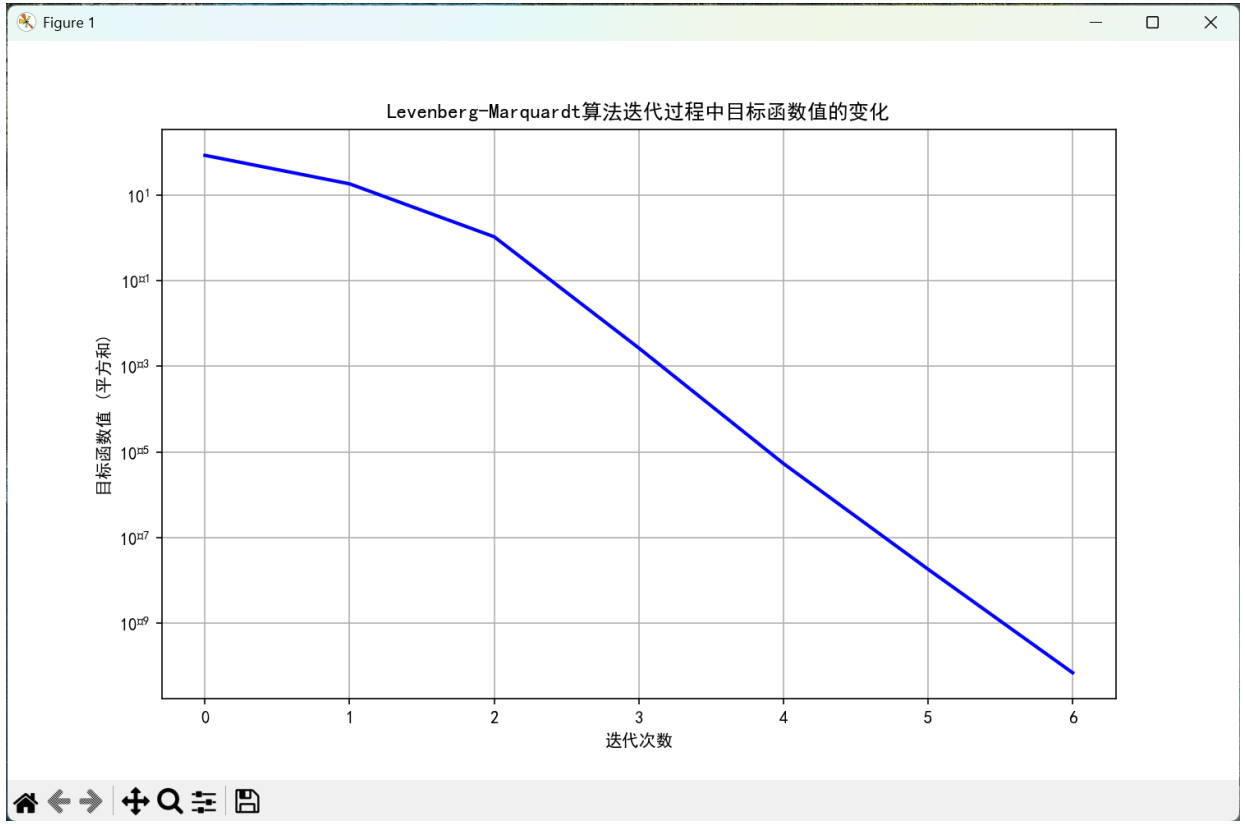
$$\mathbf{f}(x,y)=\begin{pmatrix} f_1(x,y) \\ f_2(x,y) \end{pmatrix}=\begin{pmatrix} x^2+y-11 \\ x+y^2-7 \end{pmatrix}$$

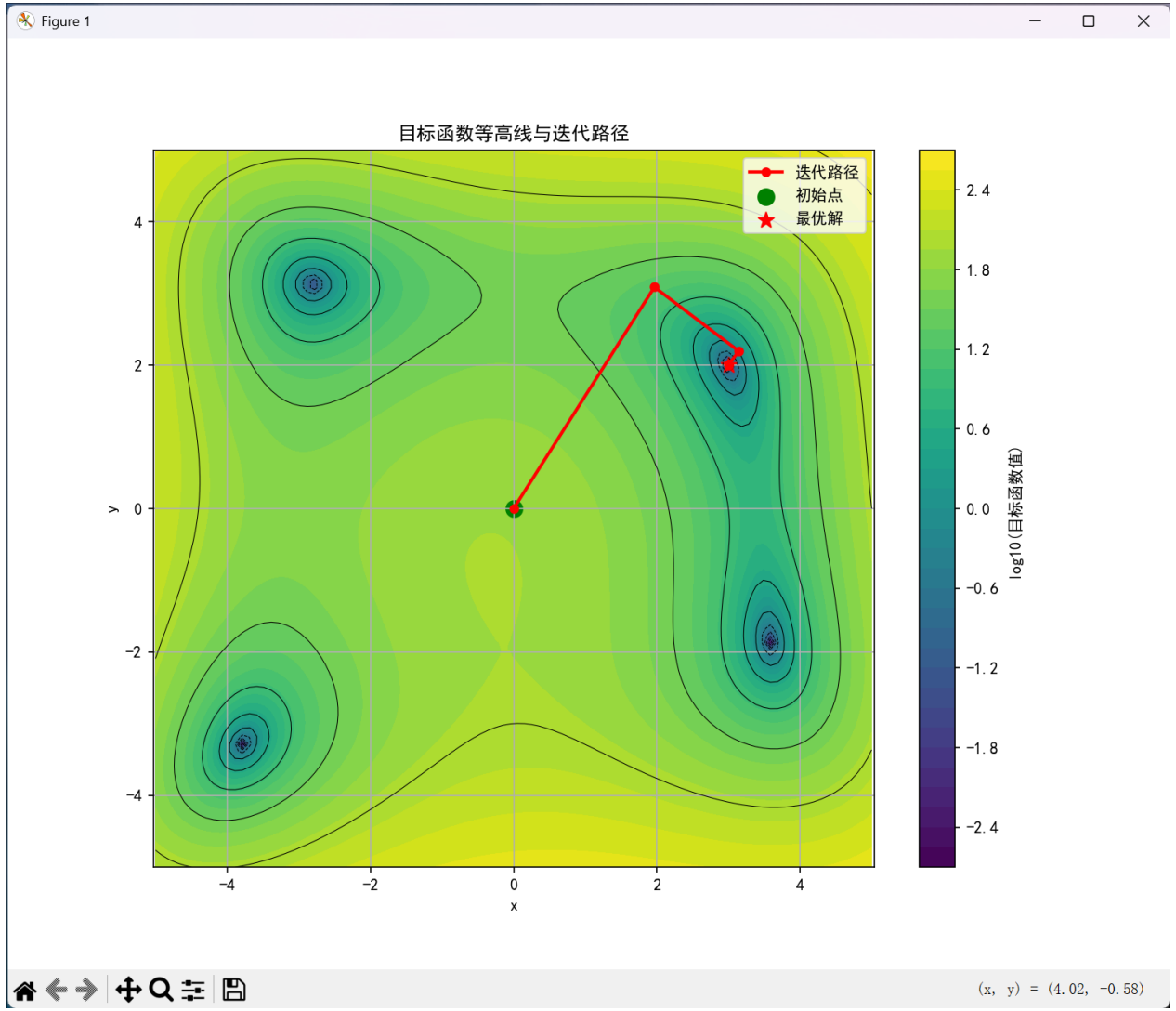
$$(x^*,y^*)=\arg\min_{x,y} f(x,y)=\frac{1}{2}\mathbf{f}^T(x,y)\mathbf{f}(x,y)=\frac{1}{2}\left(f_1^2(x,y)+f_2^2(x,y)\right)$$

$$\mathbf{J}(x,y)=\begin{pmatrix} \nabla^T f_1(x,y) \\ \nabla^T f_2(x,y) \end{pmatrix}=\begin{pmatrix} 2x & 1 \\ 1 & 2y \end{pmatrix}$$

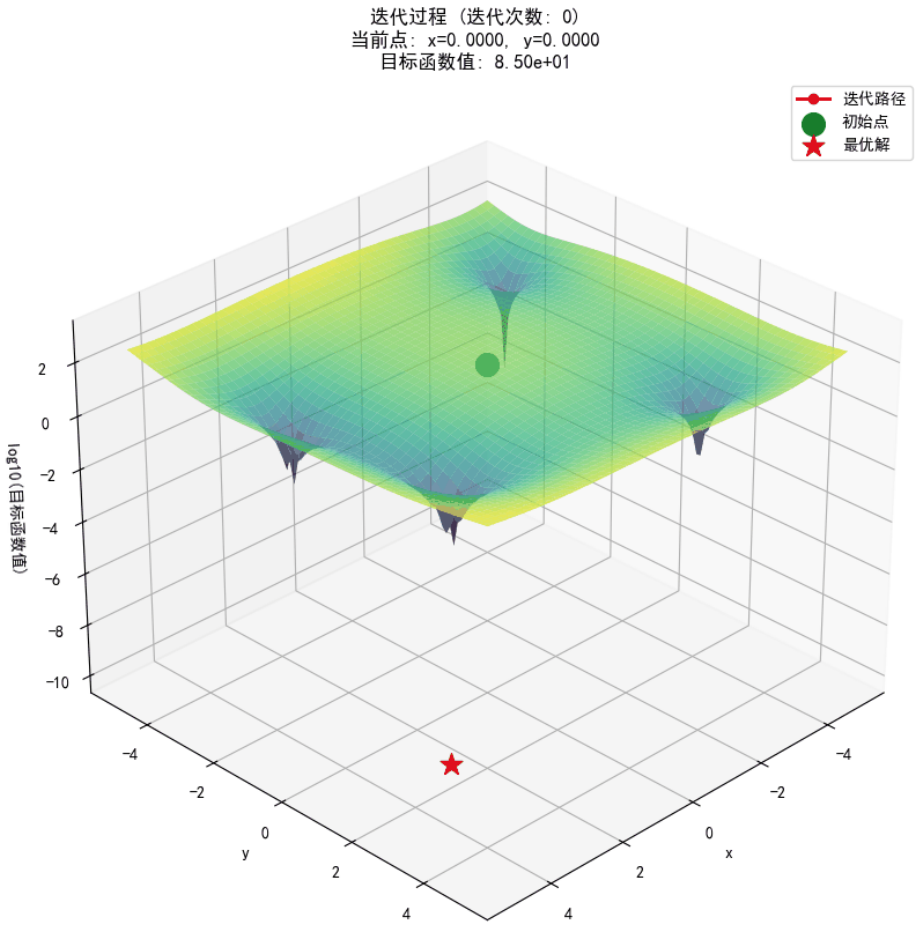
The theoretical solution to this problem is (3, 2)， and the optimal value is 0

Levenberg-Marquardt method—An example





Levenberg-Marquardt method—An example




Q+A