

---

# Notes on Visual SLAM 14 lectures – Chapter 6

## 1 Non-linear Least square problem

$$\min_x F(x) = \frac{1}{2} \|f(x)\|_2^2 \quad (1)$$

### 1.1 Analytical solution

Relatively complex, does not always exist

### 1.2 Numerical solution

- Given default value  $x_0$
- for  $k$ -th iteration, find an incremental amount  $\Delta x_k$ , so that  $\|f(x_k + \Delta x_k)\|_2^2$  becomes minimum
- if  $\Delta x_k$  is small enough, stop
- otherwise, let  $x_{k+1} = x_k + \Delta x_k$ , and return to step 2

Expand  $F(x)$  at  $X_k$ :

$$F(x_k + \Delta x_k) \approx F(x_k) + J(x_k)^T \Delta x_k + \frac{1}{2} \Delta x_k^T H(x_k) \Delta x_k \quad (2)$$

#### 1.2.1 Gradient descent

Take negative gradient as the step length

$$\frac{\partial F(x)}{\partial x} = \nabla_x F(x) \quad (3)$$

$$\Delta x = -\alpha \nabla_x F(x) \quad (4)$$

$\alpha$  : learning rate

#### 1.2.2 Second order method (Newton method)

$$\Delta x_k^* = \arg \min(F(x_k + \Delta x_k) \approx F(x_k) + J(x_k)^T \Delta x_k + \frac{1}{2} \Delta x_k^T H(x_k) \Delta x_k) \quad (5)$$

$$J + H \Delta x = 0 \implies H \Delta x = -J \quad (6)$$

$$\Delta x = -H^{-1} J \quad (7)$$

---

### 1.2.3 Gauss-Newton method

Use first-order expansion to generate second-order objective function, to avoid computing Hessian of  $F_x$

$$f(x + \Delta x) \approx f(x) + J(x)^T \Delta x \quad (8)$$

$$\Delta x^* = \arg \min_{\Delta x} \frac{1}{2} \|f(x) + J(x)^T \Delta x\|^2 \quad (9)$$

$$\begin{aligned} \frac{1}{2} \|f(x) + J(x)^T \Delta x\|_2^2 &= \frac{1}{2} (f(x) + J(x)^T \Delta x)^T (f(x) + J(x)^T \Delta x) \\ &= \frac{1}{2} (\|f(x)\|_2^2 + 2f(x)J(x)^T \Delta x + \Delta x^T J(x)J(x)^T \Delta x) \\ &= \frac{1}{2} (0 + 2 \cdot f(x) \cdot J(x) + 2 \cdot JJ^T \cdot \Delta x) \end{aligned}$$

$$J(x)f(x) + J(x)J^T(x)\Delta x = 0 \quad (10)$$

$$J(x)J^T(x)\Delta x = -f(x)J(x) \quad (11)$$

$$\Delta x^* = -(J(x)J^T(x))^{-1}f(x)J(x) \quad (12)$$

cons: numerical instability since the inverse does not always exist since  $JJ^T$  is only semi-positive definite

### 1.2.4 Levenberg-marquardt

Approximation is valid within the trust region

$$\rho = \frac{f(x + \Delta x) - f(x)}{J(x)^T \Delta x} \quad (13)$$

Numerator: actual descent in function value

Denominator: approximated descent in function value

$$\rho \begin{cases} < 1, & \text{approximated descent larger than actual descent, should shrink trust region} \\ \approx 1, & \text{approximated descent close to actual descent, trust region should remain unchanged} \\ > 1, & \text{approximated descent smaller than actual descent, should increase trust region} \end{cases} \quad (14)$$

Steps:

- for default value  $x_0$ , and default trust region radius  $\mu$
- for  $k$ -th iteration, add trust region to Gauss-Newton method:

$$\min_{\Delta x_k} \frac{1}{2} \|f(x_k) + J(x_k)^T \Delta x_k\|^2, \text{ s.t. } \|D\Delta x_k\|^2 \leq \mu \quad (15)$$

in which  $D$  is coefficient matrix, and  $\mu$  is the trust region radius

- compute  $\rho$  according to (13)
- if  $\rho > \frac{3}{4}$ , set  $\mu = 2\mu$
- if  $\rho < \frac{1}{4}$ , set  $\mu = 0.5\mu$
- if  $\rho$  is larger than certain threshold, then we can assume the approximation works, set  $x_{k+1} = x_k + \Delta x_k$
- check for convergence, if not return to step 2

---

How to solve step 2 ? Lagrange multiplier + KKT condition

$$\mathcal{L}(\Delta x_k, \lambda) = \frac{1}{2} \|f(x_k) + J(x_k)^T \Delta x_k\|^2 + \frac{\lambda}{2} (\|D\Delta x_k\|^2 - \mu) \quad (16)$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \Delta x_k} &= f(x)J(x) + J(x)J^T(x)\Delta x + \frac{\lambda}{2} \frac{\partial (D\Delta x_k)^2}{\partial \Delta x_k} \\ &= f(x)J(x) + J(x)J^T(x)\Delta x + \frac{\lambda}{2} \frac{\partial \Delta x_k^T D^T D \Delta x_k}{\partial \Delta x_k} \\ &= f(x)J(x) + J(x)J^T(x)\Delta x + \frac{\lambda}{2} \cdot 2D^T D \Delta x_k \\ &= f(x)J(x) + J(x)J^T(x)\Delta x + \lambda \cdot D^T D \Delta x_k \\ &= f(x)J(x) + (J(x)J^T(x) + \lambda D^T D)\Delta x_k = 0 \end{aligned}$$

$$\Delta x_k = -(J(x)J^T(x) + \lambda D^T D)^{-1} J(x)f(x) \quad (17)$$

$$(H + \lambda I)\Delta x_k = g \quad (18)$$

when  $\lambda$  is small, similar to Gauss-Newton method; when  $\lambda$  is large, similar to steepest descent method

### 1.2.5 Dog-leg method

$$\delta_{gn} = -(J^T J)^{-1} J^T f(x) \quad (19)$$

$$\delta_{sd} = -J^T f(x) \quad (20)$$

Linearize the objective function along the steepest descent direction:

$$\begin{aligned} F(x + t\delta_{sd}) &\approx \frac{1}{2} \|f(x) + tJ(x)\delta_{sd}\|^2 \\ &= F(x) + t\delta_{sd}^T J^T f(x) + \frac{1}{2} t^2 \|J\delta_{sd}\|^2 \end{aligned}$$

since  $F(x) = \frac{1}{2} \|f(x)\|^2$

to compute the value of the parameter  $t$  at the Cauchy point, the derivative of the last expression with respect to  $t$  is imposed to be equal to zero, giving:

$$t = -\frac{\delta_{sd}^T J^T f(x)}{\|J\delta_{sd}\|^2} = \frac{\|\delta_{sd}\|^2}{\|J\delta_{sd}\|^2} \quad (21)$$

select the update step  $\delta_k$  as equal to:

- $\delta_{gn}$  if the Gauss-Newton step is within the trust region  $\|\delta_{gn}\| \leq \Delta$
- $\frac{\Delta}{\|\delta_{sd}\|} \delta_{sd}$  if both the Gauss-Newton and the steepest descent steps are outside the trust region ( $t\|\delta_{sd}\|$ )
- $t\delta_{sd} + s(\delta_{gn} - t\delta_{sd})$  with  $s$  such that  $\|\delta\| = \Delta$  if the Gauss-Newton step is outside the trust region but the steepest descent step is inside (dog leg step)

## 2 State estimation

### 2.1 Classical SLAM problem formulation

$$\begin{cases} x_k = f(x_{k-1}, u_k) + w_k \\ z_{k,j} = h(y_i, x_k) + v_{k,j} \end{cases} \quad (22)$$

map world coordinate to camera coordinate

$$P_{uv} = \frac{K_c T_{cw} P_w}{Z_c} \quad (23)$$

---

## 2.2 Data processing

### 2.2.1 Incremental data

- only consider data under current timestamp
- could have accumulative error

### 2.2.2 Batch data

- valid across larger timestamp coverage
- takes more time

### 2.2.3 Sliding window

- only consider the most recent samples for optimization
- pro: considers both optimization time and accuracy

### 2.2.4 Frontend and backend

- frontend uses lightweight method for estimating trajectory –filter/frame matching/pre-integration of IMU/constant velocity motion assumption
- perform optimization periodically

## 2.3 State estimation

- solve  $p(x, y|z, u)$
- Use Bayes rule:  $P(x, y|z, u) \cdot P(z, u) = P(z, u|x, y) \cdot P(x, y)$

$$p(x, y|z, u) = \frac{P(z, u|x, y) \cdot P(x, y)}{P(z, u)} \quad (24)$$

MAP (maximum a posteriori)

$$(x, y)_{MAP}^* = \arg \max_{x, y} P(x, y|z, u) = \arg \max_{x, y} (P(z, u|x, y) \cdot P(x, y)) \quad (25)$$

Note: sometimes  $P(x, y)$  is not always available, so:

$$(x, y)_{MLE}^* = \arg \max_{x, y} P(x, y|z, u) \approx \arg \max_{x, y} (P(z, u|x, y)) \quad (26)$$

## 2.4 Observation error

$$z_{k,j} = h(y_j, x_k) + v_{k,j} \quad (27)$$

noise term:

$$v_k \sim \mathcal{N}(0, Q_{k,j}) \quad (28)$$

observation probability:

$$P(z_{j,k}|x_k, y_j) = N(h(y_j, x_k), Q_{k,j}) \quad (29)$$

Multivariate Gaussian distribution:

$$x \sim \mathcal{N}(\mu, \Sigma) \quad (30)$$

Univariate Gaussian:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \left( -\frac{(x - \mu)^2}{2\sigma^2} \right) \quad (31)$$

---

Multivariate Gaussian:

$$F(x) = \frac{1}{\sqrt{(2\pi)^N \det(\Sigma)}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right) \quad (32)$$

Take negative log on both sides:

$$-\ln(F(x)) = \frac{1}{2} \ln((2\pi)^N \det(\Sigma)) + \frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu) \quad (33)$$

As a result, we have:

$$\begin{aligned} (x_k, y_j)^* &= \arg \max \mathcal{N}(h(y_j, x_k), Q_{k,j}) \\ &= \arg \min ((z_{k,j} - h(x_k, y_j))^T Q_{k,j}^{-1} (z_{k,j} - h(x_k, y_j))) \end{aligned}$$

Note:  $Q_{k,j}^{-1}$  can also be called information matrix, also a measure of how confident the sensor data is (the smaller the covariance term, the bigger the corresponding term in the information matrix)

## 2.5 Batch processing

$$\begin{aligned} P(\vec{z}, \vec{u} | \vec{x}, \vec{y}) &= P(\vec{z} | \vec{x}, \vec{y}) \cdot P(\vec{u} | \vec{x}, \vec{y}) \\ &= \Pi_k P(u_k | x_{k-1}, x_k) \Pi_{k,j} P(z_{k,j} | x_k, y_j) \end{aligned}$$

since  $u_k$  only depends on  $x_{k-1}$  and  $x_k$  and similarly for  $z_{k,j}$

Define the errors for motion and observation:

$$e_{u,k} = x_k - f(x_{k-1}, u_k) \quad (34)$$

$$e_{z,j,k} = z_{k,j} - h(x_k, y_j) \quad (35)$$

we then have the updated MLE equation:

$$\min J(x, y) = \Sigma_k e_{u,k}^T R_k^{-1} e_{u,k} + \Sigma_k \Sigma_j e_{z,k,j}^T Q_{k,j}^{-1} e_{z,k,j} \quad (36)$$

## 2.6 Example

### 2.6.1 Example 1

Consider a simple discrete time system:

$$\begin{aligned} x_k &= x_{k-1} + u_k + w_k, & w_k &\sim \mathcal{N}(0, Q_k) \\ z_k &= x_k + n_k, & n_k &\sim \mathcal{N}(0, R_k) \end{aligned}$$

take  $k = 1, 2, 3, \dots$ , we have:  $x = [x_0, x_1, x_2, x_3]^T$  and batch observation value  $z = [z_1, z_2, z_3]^T$ . Let  $u = [u_1, u_2, u_3]^T$ , according to the previous derivation, we know the maximum a posteriori:

$$x_{map}^* = \Pi_{k=1}^3 P(u_k | x_{k-1}, x_k) \Pi_{k=1}^3 P(z_k | x_k) \quad (37)$$

in which  $P(u_k | x_{k-1}, x_k) = \mathcal{N}(x_k - x_{k-1}, Q_k)$  and  $P(z_k | x_k) = \mathcal{N}(x_k, R_k)$  then we would have  $e_{u,k} = u_k - (x_k - x_{k-1})$  and  $e_{z,k} = z_k - x_k$  and the objective function for least square becomes:

$$\min \Sigma_{k=1}^3 e_{u,k}^T Q_k^{-1} e_{u,k} + \Sigma_{k=1}^3 e_{z,k}^T R_k^{-1} e_{z,k} \quad (38)$$

Deem  $y = [u, z]^T$ , then we have:

$$y - Hx = e \sim \mathcal{N}(0, \Sigma) \quad (39)$$

---

in which

$$H = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (40)$$

and  $\Sigma = \text{diag}(Q_1, Q_2, Q_3, R_1, R_2, R_3)$  the problem can be written as:

$$x_{map}^* = \arg \min e^T \Sigma^{-1} e \quad (41)$$

Deem  $f(x, y) = (y - Hx)^T \Sigma^{-1} (y - Hx)$  then we have:

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{\partial(y - Hx)}{\partial x} \cdot \frac{\partial f}{\partial(y - Hx)} \\ &= -H^T \cdot 2 \cdot \Sigma^{-1} \cdot (y - Hx) \\ &= 0 \\ H^T \Sigma^{-1} y &= H^T \Sigma^{-1} H x_{map}^* \\ x_{map}^* &= (H^T \Sigma^{-1} H)^{-1} H^T \Sigma^{-1} y \end{aligned}$$

### 2.6.2 Example 2

Suppose there exists a curve  $y = \exp(ax^2 + bx + c) + w$  in which  $w \sim (0, \sigma^2)$  Now suppose we have  $N$  data points  $(x, y)$ , then we have to solve the least square problem:

$$\min_{a,b,c} \frac{1}{2} \sum_{i=1}^N \|y_i - \exp(ax_i^2 + bx_i + c)\|^2 \quad (42)$$

define error as  $e_i = y_i - \exp(ax_i^2 + bx_i + c)$ , then we can solve derivative for each variable:

$$\begin{aligned} \frac{\partial e_i}{\partial a} &= -x_i^2 \exp(ax_i^2 + bx_i + c) \\ \frac{\partial e_i}{\partial b} &= -x_i \exp(ax_i^2 + bx_i + c) \\ \frac{\partial e_i}{\partial c} &= -\exp(ax_i^2 + bx_i + c) \end{aligned}$$

so  $J_i = \begin{bmatrix} \frac{e_i}{\partial a} & \frac{e_i}{\partial b} & \frac{e_i}{\partial c} \end{bmatrix}^T$  and the equation for Gauss-Newton method becomes:

$$(\sum_{i=1}^{100} J_i(\sigma^2)^{-1} J_i^T) \Delta x_k = \sum_{i=1}^{100} -J_i(\sigma^2)^{-1} e_i \quad (43)$$