

第 3 讲基于优化的 IMU 与视觉信息融合

贺一家，高翔，崔华坤

2019 年 10 月 30 日

① 基于 Bundle Adjustment 的 VIO 融合

② 最小二乘问题的求解

基础：最速下降法，牛顿法

进阶：高斯牛顿法，LM 算法的具体实现

终极：鲁棒核函数的实现

③ VIO 残差函数的构建

视觉重投影误差

预积分模型由来及意义

预积分量方差的计算

④ 残差 Jacobian 的推导

视觉重投影残差的 Jacobian

IMU 预积分残差的雅克比

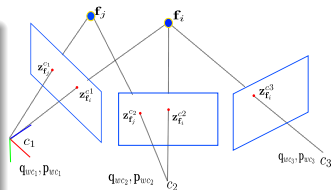
Section 1

基于 Bundle Adjustment 的 VIO 融合

视觉 SLAM 里的 Bundle Adjustment 问题

已知

- 状态量初始值：特征点的三维坐标，相机的位姿。
- 系统测量值：特征点在不同图像上的图像坐标。



问题：如何估计状态量的最优值？

解决方式

构建误差函数，利用最小二乘得到状态量的最优估计：

$$\arg \min_{\mathbf{q}, \mathbf{p}, \mathbf{f}} \sum_{i=1}^m \sum_{j=1}^n \left\| \pi(\mathbf{q}_{wc_i}, \mathbf{p}_{wc_i}, \mathbf{f}_j) - \mathbf{z}_{f_j}^{c_i} \right\|_{\Sigma_{ij}} \quad (1)$$

符号定义：

- \mathbf{q} ：旋转四元数
- \mathbf{p} ：平移向量
- \mathbf{f} ：特征点 3D 坐标
- c_i ：第 i 个相机系
- $\pi(\cdot)$ ：投影函数
- $\mathbf{z}_{f_j}^{c_i}$ ： c_i 对 f_j 的观测
- Σ_{ij} ： Σ 范数

g2o or ceres 中采用如下的求解方式，实现细节是什么？¹

Input: A vector function $f : \mathcal{R}^m \rightarrow \mathcal{R}^n$ with $n \geq m$, a measurement vector $\mathbf{x} \in \mathcal{R}^n$ and an

initial parameters estimate $\mathbf{p}_0 \in \mathcal{R}^m$.

Output: A vector $\mathbf{p}^+ \in \mathcal{R}^m$ minimizing $\|\mathbf{x} - f(\mathbf{p})\|^2$.

Algorithm:

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

$\mathbf{A} := \mathbf{J}^T \mathbf{J}; \epsilon_{\mathbf{p}} := \mathbf{x} - f(\mathbf{p}); \mathbf{g} := \mathbf{J}^T \epsilon_{\mathbf{p}};$

stop: $(\|\mathbf{g}\|_{\infty} \leq \varepsilon_1); \mu := \tau * \max_{i=1, \dots, m} (A_{ii});$

while (not stop) and $(k < k_{max})$

$k := k + 1;$

repeat

Solve $(\mathbf{A} + \mu \mathbf{I}) \delta_{\mathbf{p}} = \mathbf{g};$

if $(\|\delta_{\mathbf{p}}\| \leq \varepsilon_2 (\|\mathbf{p}\| + \varepsilon_2))$

stop:=true;

else

$\mathbf{p}_{new} := \mathbf{p} + \delta_{\mathbf{p}};$

$\rho := (\|\epsilon_{\mathbf{p}}\|^2 - \|\mathbf{x} - f(\mathbf{p}_{new})\|^2) / (\delta_{\mathbf{p}}^T (\mu \delta_{\mathbf{p}} + \mathbf{g}));$

if $\rho > 0$

stop: $(\|\epsilon_{\mathbf{p}}\| - \|\mathbf{x} - f(\mathbf{p}_{new})\| < \varepsilon_4 \|\epsilon_{\mathbf{p}}\|);$

$\mathbf{p} = \mathbf{p}_{new};$

$\mathbf{A} := \mathbf{J}^T \mathbf{J}; \epsilon_{\mathbf{p}} := \mathbf{x} - f(\mathbf{p}); \mathbf{g} := \mathbf{J}^T \epsilon_{\mathbf{p}};$

stop: (stop) or $(\|\mathbf{g}\|_{\infty} \leq \varepsilon_1);$

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

else

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

endif

endif

until $(\rho > 0)$ or (stop)

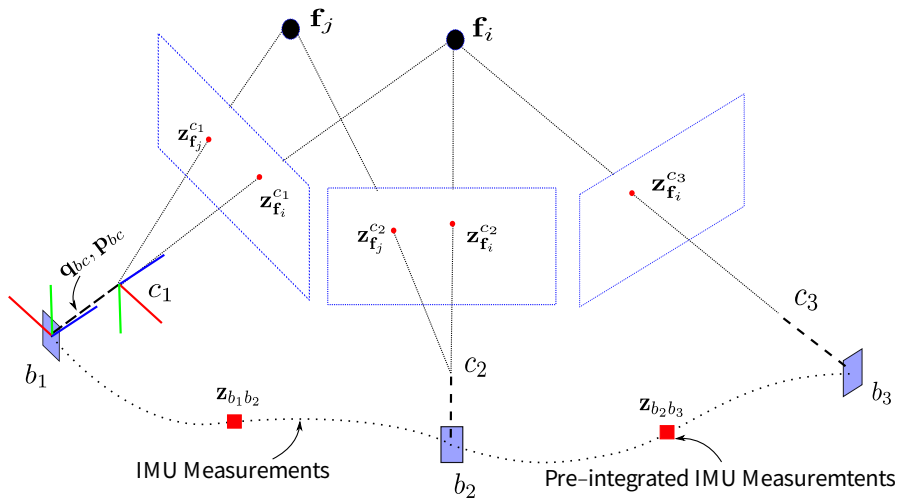
stop: $(\|\epsilon_{\mathbf{p}}\| \leq \varepsilon_3);$

endwhile

$\mathbf{p}^+ := \mathbf{p};$

¹本页数学符号和网页无关

VIO 信息融合问题



如何构建 IMU 误差 $z_{b_1b_2}$? 如何设定多个信息源权重 ? 如何求解 ?

Section 2

最小二乘问题的求解

最小二乘基础概念

定义

找到一个 n 维的变量 $\mathbf{x}^* \in \mathbb{R}^n$ ，使得损失函数 $F(\mathbf{x})$ 取局部最小值：

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

其中 f_i 是残差函数，比如测量值和预测值之间的差，且有 $m \geq n$ 。局部最小值指对任意 $\|\mathbf{x} - \mathbf{x}^*\| < \delta$ 有 $F(\mathbf{x}^*) \leq F(\mathbf{x})$

损失函数泰勒展开

假设损失函数 $F(\mathbf{x})$ 是可导并且平滑的，因此，二阶泰勒展开：

$$F(\mathbf{x} + \Delta\mathbf{x}) = F(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\top \mathbf{H}\Delta\mathbf{x} + O(\|\Delta\mathbf{x}\|^3) \quad (2)$$

其中 \mathbf{J} 和 \mathbf{H} 分别为损失函数 F 对变量 \mathbf{x} 的一阶导和二阶导矩阵。

损失函数泰勒展开性质

忽略泰勒展开的高阶项，损失函数变成了二次函数，可以轻易得到如下性质：

- 如果在点 x_s 处有导数为 0，则称这个点为稳定点。
- 在点 x_s 处对应的 Hessian 为 H ：
- 如果是正定矩阵，即它的特征值都大于 0，则在 x_s 处有 $F(x)$ 为局部最小值；
- 如果是负定矩阵，即它的特征值都小于 0，则在 x_s 处有 $F(x)$ 为局部最大值；
- 如果是不定矩阵，即它的特征值大于 0 也有小于 0 的，则 x_s 处为鞍点。

求解法

- 直接求解：线性最小二乘。
- 迭代下降法：适用于线性和非线性最小二乘。

迭代下降法求解：下降法

迭代法初衷

找一个下降方向使损失函数随 \mathbf{x} 的迭代逐渐减小，直到 \mathbf{x} 收敛到 \mathbf{x}^* ：

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

分两步：第一，找下降方向单位向量 \mathbf{d} ，第二，确定下降步长 α 。

假设 α 足够小，我们可以对损失函数 $F(\mathbf{x})$ 进行一阶泰勒展开：

$$F(\mathbf{x} + \alpha \mathbf{d}) \approx F(\mathbf{x}) + \alpha \mathbf{J} \mathbf{d}$$

只需寻找下降方向，满足：

$$\mathbf{J} \mathbf{d} < 0$$

通过 line search 方法找到下降的步长： $\alpha^* = \operatorname{argmin}_{\alpha > 0} \{F(\mathbf{x} + \alpha \mathbf{d})\}$

最速下降法和牛顿法

最速下降法：适用于迭代的开始阶段

从下降方向的条件可知： $\mathbf{Jd} = \|\mathbf{J}\| \cos \theta$ ， θ 表示下降方向和梯度方向的夹角。当 $\theta = \pi$ ，有

$$\mathbf{d} = \frac{-\mathbf{J}^\top}{\|\mathbf{J}\|}$$

即 **梯度的负方向**为最速下降方向。缺点：最优值附近震荡，收敛慢。

牛顿法：适用于最优值附近

在局部最优点 \mathbf{x}^* 附近，如果 $\mathbf{x} + \Delta\mathbf{x}$ 是最优解，则损失函数对 $\Delta\mathbf{x}$ 的导数等于 0，对公式 (2) 取一阶导有：

$$\frac{\partial}{\partial \Delta\mathbf{x}} \left(F(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\top \mathbf{H}\Delta\mathbf{x} \right) = \mathbf{J}^\top + \mathbf{H}\Delta\mathbf{x} = 0 \quad (3)$$

得到： $\Delta\mathbf{x} = -\mathbf{H}^{-1}\mathbf{J}^\top$ 。缺点：二阶导矩阵计算复杂。

阻尼法

Damp Method

将损失函数的二阶泰勒展开记作

$$F(\mathbf{x} + \Delta\mathbf{x}) \approx L(\Delta\mathbf{x}) \equiv F(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^\top \mathbf{H}\Delta\mathbf{x}$$

求以下函数的最小化：

$$\Delta\mathbf{x} \equiv \arg \min_{\Delta\mathbf{x}} \left\{ L(\Delta\mathbf{x}) + \frac{1}{2}\mu\Delta\mathbf{x}^\top \Delta\mathbf{x} \right\}$$

其中， $\mu \geq 0$ 为阻尼因子， $\frac{1}{2}\mu\Delta\mathbf{x}^\top \Delta\mathbf{x} = \frac{1}{2}\mu\|\Delta\mathbf{x}\|^2$ 是惩罚项。
对新的损失函数求一阶导，并令其等于 0 有：

$$\begin{aligned} \mathbf{L}'(\Delta\mathbf{x}) + \mu\Delta\mathbf{x} &= \mathbf{0} \\ \Rightarrow (\mathbf{H} + \mu\mathbf{I}) \Delta\mathbf{x} &= -\mathbf{J}^\top \end{aligned} \tag{4}$$

非线性最小二乘

符号说明

为了公式约简，可将残差组合成向量的形式。

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \dots \\ f_m(\mathbf{x}) \end{bmatrix} \quad (5)$$

则有： $\mathbf{f}^\top(\mathbf{x})\mathbf{f}(\mathbf{x}) = \sum_{i=1}^m (f_i(\mathbf{x}))^2$

同理，如果记 $\mathbf{J}_i(\mathbf{x}) = \frac{\partial f_i(\mathbf{x})}{\partial \mathbf{x}}$ 则有：

$$\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{J} = \begin{bmatrix} \mathbf{J}_1(\mathbf{x}) \\ \dots \\ \mathbf{J}_m(\mathbf{x}) \end{bmatrix}, \quad \mathbf{J}_i(\mathbf{x}) = \begin{bmatrix} \frac{\partial f_i(\mathbf{x})}{\partial x_1} & \frac{\partial f_i(\mathbf{x})}{\partial x_2} & \dots \end{bmatrix} \quad (6)$$

非线性最小二乘

基础

残差函数 $f(\mathbf{x})$ 为非线性函数，对其一阶泰勒近似有：

$$\mathbf{f}(\mathbf{x} + \Delta\mathbf{x}) \approx \ell(\Delta\mathbf{x}) \equiv \mathbf{f}(\mathbf{x}) + \mathbf{J}\Delta\mathbf{x}$$

请特别注意，这里的 \mathbf{J} 是残差函数 \mathbf{f} 的雅克比矩阵。代入损失函数：

$$\begin{aligned} F(\mathbf{x} + \Delta\mathbf{x}) &\approx L(\Delta\mathbf{x}) \equiv \frac{1}{2} \ell(\Delta\mathbf{x})^\top \ell(\Delta\mathbf{x}) \\ &= \frac{1}{2} \mathbf{f}^\top \mathbf{f} + \Delta\mathbf{x}^\top \mathbf{J}^\top \mathbf{f} + \frac{1}{2} \Delta\mathbf{x}^\top \mathbf{J}^\top \mathbf{J} \Delta\mathbf{x} \quad (7) \\ &= F(\mathbf{x}) + \Delta\mathbf{x}^\top \mathbf{J}^\top \mathbf{f} + \frac{1}{2} \Delta\mathbf{x}^\top \mathbf{J}^\top \mathbf{J} \Delta\mathbf{x} \end{aligned}$$

这样损失函数就近似成了一个二次函数，并且如果雅克比是满秩的，则 $\mathbf{J}^\top \mathbf{J}$ 正定，损失函数有最小值。

另外，易得： $F'(\mathbf{x}) = (\mathbf{J}^\top \mathbf{f})^\top$ ，以及 $F''(\mathbf{x}) \approx \mathbf{J}^\top \mathbf{J}$ 。

Gauss-Newton 和 LM

Gauss-Newton Method

令公式 (7) 的一阶导等于 0，得到：

$$(\mathbf{J}^\top \mathbf{J}) \Delta \mathbf{x}_{\text{gn}} = -\mathbf{J}^\top \mathbf{f} \quad (8)$$

上式就是通常论文里看到的 $\mathbf{H} \Delta \mathbf{x}_{\text{gn}} = \mathbf{b}$ ，称其为 normal equation.

The Levenberg-Marquardt Method

Levenberg (1944) 和 Marquardt (1963) 先后对高斯牛顿法进行了改进，求解过程中引入了阻尼因子：

$$(\mathbf{J}^\top \mathbf{J} + \mu \mathbf{I}) \Delta \mathbf{x}_{\text{lm}} = -\mathbf{J}^\top \mathbf{f} \quad \text{with } \mu \geq 0$$

疑问：LM 中阻尼因子有什么作用，它怎么设定呢？

阻尼因子的作用

- $\mu > 0$ 保证 $(\mathbf{J}^\top \mathbf{J} + \mu \mathbf{I})$ 正定，迭代朝着下降方向进行。
- μ 非常大，则 $\Delta \mathbf{x}_{lm} = -\frac{1}{\mu} \mathbf{J}^\top \mathbf{f} = -\frac{1}{\mu} F'(\mathbf{x})^\top$ ，接近最速下降法。
- μ 比较小，则 $\Delta \mathbf{x}_{lm} \approx \Delta \mathbf{x}_{gn}$ ，接近高斯牛顿法。

阻尼因子初始值的选取

阻尼因子 μ 大小是相对于 $\mathbf{J}^\top \mathbf{J}$ 的元素而言的。半正定的信息矩阵 $\mathbf{J}^\top \mathbf{J}$ 特征值 $\{\lambda_j\}$ 和对应的特征向量为 $\{\mathbf{v}_j\}$ 。对 $\mathbf{J}^\top \mathbf{J}$ 做特征值分解分解后有： $\mathbf{J}^\top \mathbf{J} = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top$ 可得：

$$\Delta \mathbf{x}_{lm} = - \sum_{j=1}^n \frac{\mathbf{v}_j^\top \mathbf{F}'^\top}{\lambda_j + \mu} \mathbf{v}_j \quad (9)$$

所以，一个简单的 μ_0 初始值的策略就是：

$$\mu_0 = \tau \cdot \max \left\{ \left(\mathbf{J}^\top \mathbf{J} \right)_{ii} \right\}$$

通常，按需设定 $\tau \sim [10^{-8}, 1]$ 。

阻尼因子 μ 的更新策略

定性分析，直观感受阻尼因子的更新：

- ① 如果 $\Delta \mathbf{x} \rightarrow F(\mathbf{x}) \uparrow$ ，则 $\mu \uparrow \rightarrow \Delta \mathbf{x} \downarrow$ ，增大阻尼减小步长，拒绝本次迭代。
- ② 如果 $\Delta \mathbf{x} \rightarrow F(\mathbf{x}) \downarrow$ ，则 $\mu \downarrow \rightarrow \Delta \mathbf{x} \uparrow$ ，减小阻尼增大步长。加快收敛，减少迭代次数。

定量分析，阻尼因子更新策略通过比例因子来确定的：

$$\rho = \frac{F(\mathbf{x}) - F(\mathbf{x} + \Delta \mathbf{x}_{lm})}{L(\mathbf{0}) - L(\Delta \mathbf{x}_{lm})} \quad (10)$$

其中：

$$\begin{aligned}
 L(\mathbf{0}) - L(\Delta \mathbf{x}_{lm}) &= -\Delta \mathbf{x}_{lm}^T \mathbf{J}^T \mathbf{f} - \frac{1}{2} \Delta \mathbf{x}_{lm}^T \mathbf{J}^T \mathbf{J} \Delta \mathbf{x}_{lm} \\
 &\stackrel{\mathbf{b} = -\mathbf{J}^T \mathbf{f}}{=} -\frac{1}{2} \Delta \mathbf{x}_{lm}^T \left(-2\mathbf{b} + \underbrace{(\mathbf{J}^T \mathbf{J} + \mu \mathbf{I} - \mu \mathbf{I})}_{(J'J + \mu I)dx = -J'f} \Delta \mathbf{x}_{lm} \right) \\
 &= \frac{1}{2} \Delta \mathbf{x}_{lm}^T (\mu \Delta \mathbf{x}_{lm} + \mathbf{b})
 \end{aligned} \quad (11)$$

Marquardt 策略

首先比例因子分母始终大于 0，如果：

- $\rho < 0$, 则 $F(\mathbf{x}) \uparrow$, 应该 $\mu \uparrow \rightarrow \Delta \mathbf{x} \downarrow$, 增大阻尼减小步长。
- 如果 $\rho > 0$ 且比较大, 减小 μ , 让 LM 接近 Gauss-Newton 使得系统更快收敛。
- 反之, 如果是比较小的正数, 则增大阻尼 μ , 缩小迭代步长。

1963 年 Marquardt 提出了一个如下的阻尼策略:

$$\begin{aligned} & \text{if } \rho < 0.25 \\ & \quad \mu := \mu * 2 \\ & \text{elseif } \rho > 0.75 \\ & \quad \mu := \mu / 3 \end{aligned} \tag{12}$$

Marquardt 好不好呢？如下图所示²：



²Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff. "Methods for non-linear least squares problems". [Lecture Notes in Computer Science](#) (1999)

Nielsen 策略 (被 g2o, ceres 采用)

if $\rho > 0$

$$\mu := \mu * \max \left\{ \frac{1}{3}, 1 - (2\rho - 1)^3 \right\}; \quad \nu := 2 \quad (13)$$

else

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



鲁棒核函数的实现

引言：最小二乘中遇到 outlier 怎么处理？核函数如何在代码中实现？有多种方法³，这里主要介绍 g2o 和 ceres 中使用的 Triggs Correction⁴。

Triggs Correction

鲁棒核函数直接作用残差 $f_k(\mathbf{x})$ 上，最小二乘函数变成了如下形式：

$$\min_{\mathbf{x}} \frac{1}{2} \sum_k \rho(\|f_k(\mathbf{x})\|^2)$$

将误差的平方项记作 $s_k = \|f_k(\mathbf{x})\|^2$ ，则鲁棒核误差函数进行二阶泰勒展开有：

$$\frac{1}{2} \rho(s) = \frac{1}{2} (\text{const} + \rho' \Delta s + \frac{1}{2} \rho'' \Delta^2 s) \quad (14)$$

³Christopher Zach. "Robust bundle adjustment revisited". In: *European Conference on Computer Vision*. Springer. 2014, pp. 772–787.

⁴Bill Triggs et al. "Bundle adjustment—a modern synthesis". In: *International workshop on vision algorithms*. Springer. 1999, pp. 298–372.

Triggs Correction

上述函数中 Δs_k 的计算稍微复杂一点：

$$\begin{aligned}
 \Delta s_k &= \|f_k(\mathbf{x} + \Delta \mathbf{x})\|^2 - \|f_k(\mathbf{x})\|^2 \\
 &\approx \|f_k + \mathbf{J}_k \Delta \mathbf{x}\|^2 - \|f_k(\mathbf{x})\|^2 \\
 &= 2f_k^\top \mathbf{J}_k \Delta \mathbf{x} + (\Delta \mathbf{x})^\top \mathbf{J}_k^\top \mathbf{J}_k \Delta \mathbf{x}
 \end{aligned} \tag{15}$$

公式(15)代入公式 (14) 有：

$$\begin{aligned}
 \frac{1}{2}\rho(s) &\approx \frac{1}{2}(\rho'[2f_k^\top \mathbf{J}_k \Delta \mathbf{x} + (\Delta \mathbf{x})^\top \mathbf{J}_k^\top \mathbf{J}_k \Delta \mathbf{x}] \\
 &\quad + \frac{1}{2}\rho''[2f_k^\top \mathbf{J}_k \Delta \mathbf{x} + (\Delta \mathbf{x})^\top \mathbf{J}_k^\top \mathbf{J}_k \Delta \mathbf{x}]^2 + \text{const}) \\
 &\approx \rho' f_k^\top \mathbf{J}_k \Delta \mathbf{x} + \frac{1}{2}\rho'(\Delta \mathbf{x})^\top \mathbf{J}_k^\top \mathbf{J}_k \Delta \mathbf{x} + \rho''(\Delta \mathbf{x})^\top \mathbf{J}_k^\top f_k f_k^\top \mathbf{J}_k \Delta \mathbf{x} + \text{const} \\
 &= \rho' f_k^\top \mathbf{J}_k \Delta \mathbf{x} + \frac{1}{2}(\Delta \mathbf{x})^\top \mathbf{J}_k^\top (\rho' I + 2\rho'' f_k f_k^\top) \mathbf{J}_k \Delta \mathbf{x} + \text{const}
 \end{aligned} \tag{16}$$

Triggs Correction

对公式(16)求和后，对变量 $\Delta \mathbf{x}$ 求导，令其等于 0，得到：

$$\begin{aligned} \sum_k \mathbf{J}_k^\top (\rho' I + 2\rho'' f_k f_k^\top) \mathbf{J}_k \Delta \mathbf{x} &= - \sum_k \rho' \mathbf{J}_k^\top f_k \\ \sum_k \mathbf{J}_k^\top \mathbf{W}_k \mathbf{J}_k \Delta \mathbf{x} &= - \sum_k \rho' \mathbf{J}_k^\top f_k \end{aligned} \quad (17)$$

Example: Cauchy Cost Function

柯西鲁棒核函数的定义为：

$$\rho(s) = c^2 \log(1 + \frac{s^2}{c^2})$$

其中 c 为控制参数。对 s 的一阶导和二阶导为：

$$\rho'(s) = \frac{1}{1 + \frac{s^2}{c^2}}, \quad \rho''(s) = -\frac{1}{c^2} (\rho'(s))^2$$

g2o 代码样例

```

Vector3D rho; // 用来保存鲁棒核函数，一阶导，二阶导
// rho[0] = rho(sq_norm),
// rho[1] = rho'(sq_norm),
// rho[2] = rho''(sq_norm),
this->robustKernel()->robustify(error, rho);
InformationType weightedOmega = this->robustInformation(rho);
omega_r *= rho[1]; // 公式中的  $\rho'(r^2) * r$ 

from->b().noalias() += A.transpose() * omega_r; // 公式中的  $b = -\rho'(r^2) * r^AT * J$ 
from->A().noalias() += A.transpose() * weightedOmega * A; // 公式中的  $J^AT * W * J$ 

```

上述代码片段，基本和前面的推导一致，其中 robustInformation() 函数在 base_edge.h 中进行了实现，具体代码如下：

```

InformationType robustInformation(const Vector3D& rho)
{
    // _information 可以看成是单位矩阵
    InformationType result = rho[1] * _information;
    // 计算权重  $W = \rho' + 2 * \rho'' * r * r^AT$ 
    // 但是不知道为啥作者注释了后面这小段代码，也就是变成了  $W = \rho'$ 
    // ErrorVector weightedError = _information * _error;
    // result.noalias() += 2 * rho[2] * (weightedError * weightedError.transpose());
    return result;
}

```

回顾最小二乘求解

步骤

- 1 找到一个合适的关于状态量 x 的残差函数 $f_i(x)$ ，后续用 r, err 等表示。
- 2 计算残差函数对状态量 x 的雅克比 J 。
- 3 选定 cost function 以及其参数。
- 4 LM 算法求解。

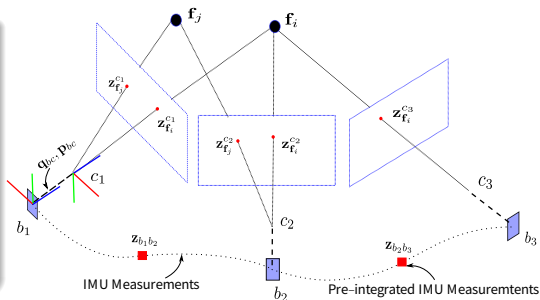
Section 3

VIO 残差函数的构建

带权重 (方差) 的残差计算

$$\begin{aligned}
 r &= \|\mathbf{f}(\mathbf{x})\|_{\Sigma}^2 \\
 &= \mathbf{f}^{\top}(\mathbf{x}) \Sigma^{-1} \mathbf{f}(\mathbf{x}) \quad (18)
 \end{aligned}$$

其中, $\mathbf{f}(\mathbf{x})$ 服从高斯分布, 协方差为 Σ 。



基于滑动窗口的 VIO Bundle Adjustment

$$\begin{aligned}
 \min_{\mathcal{X}} & \underbrace{\rho\left(\|\mathbf{r}_p - \mathbf{J}_p \mathcal{X}\|_{\Sigma_p}^2\right)}_{\text{prior}} + \underbrace{\sum_{i \in B} \rho\left(\|\mathbf{r}_b(\mathbf{z}_{b_i b_{i+1}}, \mathcal{X})\|_{\Sigma_{b_i b_{i+1}}}^2\right)}_{\text{IMU error}} \\
 & + \underbrace{\sum_{(i,j) \in F} \rho\left(\|\mathbf{r}_f(\mathbf{z}_{f_j}^{c_i}, \mathcal{X})\|_{\Sigma_{f_j}^{c_i}}^2\right)}_{\text{image error}} \quad (19)
 \end{aligned}$$

系统需要优化的状态量

为了节约计算量采用滑动窗口形式的 Bundle Adjustment, 在 i 时刻, 滑动窗口内待优化的系统状态量定义如下:

$$\begin{aligned}\mathcal{X} &= [\mathbf{x}_n, \mathbf{x}_{n+1}, \dots, \mathbf{x}_{n+N}, \lambda_m, \lambda_{m+1}, \dots, \lambda_{m+M}] \\ \mathbf{x}_i &= [\mathbf{p}_{wb_i}, \mathbf{q}_{wb_i}, \mathbf{v}_i^w, \mathbf{b}_a^{b_i}, \mathbf{b}_g^{b_i}]^\top, i \in [n, n+N]\end{aligned}\quad (20)$$

其中:

- \mathbf{x}_i 包含 i 时刻 IMU 机体的在惯性坐标系中的位置, 速度, 姿态, 以及 IMU 机体坐标系中的加速度和角速度的偏置量估计。
- n, m 分别是机体状态量, 路标在滑动窗口里的起始时刻。
- N 滑动窗口中关键帧数量。
- M 是被滑动窗口内所有关键帧观测到的路标数量。

视觉重投影误差

视觉重投影误差

定义：一个特征点在归一化相机坐标系下的估计值与观测值的差。

$$\mathbf{r}_c = \begin{bmatrix} \frac{x}{z} - u \\ \frac{y}{z} - v \end{bmatrix} \quad (21)$$

其中，待估计的状态量为特征点的三维空间坐标 $(x, y, z)^\top$ ，观测值 $(u, v)^\top$ 为特征在相机归一化平面的坐标。

逆深度参数化

特征点在归一化相机坐标系与在相机坐标系下的坐标关系为：

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \frac{1}{\lambda} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (22)$$

其中 $\lambda = 1/z$ 称为逆深度。

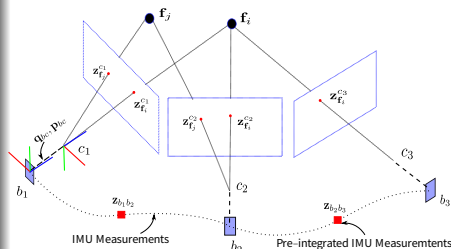
VIO 中基于逆深度的重投影误差

特征点逆深度在第 i 帧中初始化得到，在第 j 帧又被观测到，预测其在第 j 中的坐标为：

$$\begin{bmatrix} x_{c_j} \\ y_{c_j} \\ z_{c_j} \\ 1 \end{bmatrix} = \mathbf{T}_{bc}^{-1} \mathbf{T}_{wb_j}^{-1} \mathbf{T}_{wb_i} \mathbf{T}_{bc} \begin{bmatrix} \frac{1}{\lambda} u_{c_i} \\ \frac{1}{\lambda} v_{c_i} \\ \frac{1}{\lambda} \\ 1 \end{bmatrix} \quad (23)$$

视觉重投影误差为：

$$\mathbf{r}_c = \begin{bmatrix} \frac{x_{c_j}}{z_{c_j}} - u_{c_j} \\ \frac{y_{c_j}}{z_{c_j}} - v_{c_j} \end{bmatrix} \quad (24)$$



IMU 测量值积分

IMU 的真实值为 ω, \mathbf{a} , 测量值为 $\tilde{\omega}, \tilde{\mathbf{a}}$, 则有:

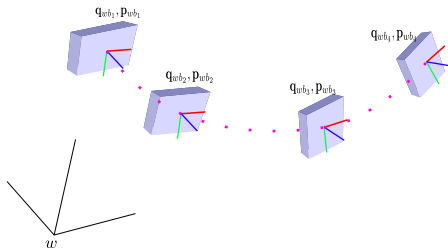
$$\tilde{\omega}^b = \omega^b + \mathbf{b}^g + \mathbf{n}^g \quad (25)$$

$$\tilde{\mathbf{a}}^b = \mathbf{q}_{bw}(\mathbf{a}^w + \mathbf{g}^w) + \mathbf{b}^a + \mathbf{n}^a \quad (26)$$

上标 g 表示 gyro, a 表示 acc, w 表示在世界坐标系 world, b 表示 imu 机体坐标系 body。

PVQ 对时间的导数可写成:

$$\begin{aligned} \dot{\mathbf{p}}_{wb_t} &= \mathbf{v}_t^w \\ \dot{\mathbf{v}}_t^w &= \mathbf{a}_t^w \\ \dot{\mathbf{q}}_{wb_t} &= \mathbf{q}_{wb_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\omega^{b_t} \end{bmatrix} \end{aligned} \quad (27)$$



从第 i 时刻的 PVQ 对 IMU 的测量值进行积分得到第 j 时刻的 PVQ:

$$\begin{aligned}
 \mathbf{p}_{wb_j} &= \mathbf{p}_{wb_i} + \mathbf{v}_i^w \Delta t + \int \int_{t \in [i, j]} (\mathbf{q}_{wb_t} \mathbf{a}^{b_t} - \mathbf{g}^w) \delta t^2 \\
 \mathbf{v}_j^w &= \mathbf{v}_i^w + \int_{t \in [i, j]} (\mathbf{q}_{wb_t} \mathbf{a}^{b_t} - \mathbf{g}^w) \delta t \\
 \mathbf{q}_{wb_j} &= \int_{t \in [i, j]} \mathbf{q}_{wb_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2} \boldsymbol{\omega}^{b_t} \end{bmatrix} \delta t
 \end{aligned} \tag{28}$$

在迭代寻优时，每次对 \mathbf{q}_{wb_t} 的更新都需要重新计算一下 (28)

问题：每次 \mathbf{q}_{wb_t} 优化更新后，都需要重新进行积分，运算量较大。

IMU 预积分

一个很简单的公式转换，就可以将积分模型转为预积分模型：

$$\mathbf{q}_{wb_t} = \mathbf{q}_{wb_i} \otimes \mathbf{q}_{b_i b_t} \quad (29)$$

那么，PVQ 积分公式中的积分项则变成相对于第 i 时刻的姿态，而不是相对于世界坐标系的姿态：

$$\begin{aligned} \mathbf{p}_{wb_j} &= \mathbf{p}_{wb_i} + \mathbf{v}_i^w \Delta t - \frac{1}{2} \mathbf{g}^w \Delta t^2 + \underline{\mathbf{q}_{wb_i}} \int \int_{t \in [i, j]} (\mathbf{q}_{b_i b_t} \mathbf{a}^{b_t}) \delta t^2 \\ \mathbf{v}_j^w &= \mathbf{v}_i^w - \mathbf{g}^w \Delta t + \underline{\mathbf{q}_{wb_i}} \int_{t \in [i, j]} (\mathbf{q}_{b_i b_t} \mathbf{a}^{b_t}) \delta t \\ \mathbf{q}_{wb_j} &= \underline{\mathbf{q}_{wb_i}} \int_{t \in [i, j]} \mathbf{q}_{b_i b_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2} \boldsymbol{\omega}^{b_t} \end{bmatrix} \delta t \end{aligned} \quad (30)$$

对 \mathbf{q}_{wb_i} 更新不会影响积分值

预积分量

预积分量仅仅跟 IMU 测量值有关，它将一段时间内的 IMU 数据直接积分起来就得到了**预积分量**：

$$\begin{aligned}
 \alpha_{b_i b_j} &= \int \int_{t \in [i, j]} (\mathbf{q}_{b_i b_t} \mathbf{a}^{b_t}) \delta t^2 \\
 \beta_{b_i b_j} &= \int_{t \in [i, j]} (\mathbf{q}_{b_i b_t} \mathbf{a}^{b_t}) \delta t \\
 \mathbf{q}_{b_i b_j} &= \int_{t \in [i, j]} \mathbf{q}_{b_i b_t} \otimes \begin{bmatrix} 0 \\ \frac{1}{2} \boldsymbol{\omega}^{b_t} \end{bmatrix} \delta t
 \end{aligned} \tag{31}$$

重新整理下 PVQ 的积分公式，有：

$$\begin{bmatrix} \mathbf{p}_{wb_j} \\ \mathbf{v}_j^w \\ \mathbf{q}_{wb_j} \\ \mathbf{b}_j^a \\ \mathbf{b}_j^g \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{wb_i} + \mathbf{v}_i^w \Delta t - \frac{1}{2} \mathbf{g}^w \Delta t^2 + \mathbf{q}_{wb_i} \alpha_{b_i b_j} \\ \mathbf{v}_i^w - \mathbf{g}^w \Delta t + \mathbf{q}_{wb_i} \beta_{b_i b_j} \\ \mathbf{q}_{wb_i} \mathbf{q}_{b_i b_j} \\ \mathbf{b}_i^a \\ \mathbf{b}_i^g \end{bmatrix} \tag{32}$$

IMU 的预积分误差

预积分误差

定义：一段时间内 IMU 构建的预积分量作为测量值，对两时刻之间的状态量进行约束，

$$\begin{bmatrix} \mathbf{r}_p \\ \mathbf{r}_q \\ \mathbf{r}_v \\ \mathbf{r}_{ba} \\ \mathbf{r}_{bg} \end{bmatrix}_{15 \times 1} = \begin{bmatrix} \mathbf{q}_{b_i w} (\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2} \mathbf{g}^w \Delta t^2) - \boldsymbol{\alpha}_{b_i b_j} \\ 2[\mathbf{q}_{b_j b_i} \otimes (\mathbf{q}_{b_i w} \otimes \mathbf{q}_{wb_j})]_{xyz} \\ \mathbf{q}_{b_i w} (\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t) - \boldsymbol{\beta}_{b_i b_j} \\ \mathbf{b}_j^a - \mathbf{b}_i^a \\ \mathbf{b}_j^g - \mathbf{b}_i^g \end{bmatrix} \quad (33)$$

上面误差中位移，速度，偏置都是直接相减得到。第二项是关于四元数的旋转误差，其中 $[\cdot]_{xyz}$ 表示只取四元数的虚部 (x, y, z) 组成的三维向量。

预积分的离散形式

这里使用 mid-point 方法，即两个相邻时刻 k 到 $k+1$ 的位姿是用两个时刻的测量值 \mathbf{a}, ω 的平均值来计算：

$$\begin{aligned}
 \omega &= \frac{1}{2}((\omega^{b_k} - \mathbf{b}_k^g) + (\omega^{b_{k+1}} - \mathbf{b}_k^g)) \\
 \mathbf{q}_{b_i b_{k+1}} &= \mathbf{q}_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix} \\
 \mathbf{a} &= \frac{1}{2}(\mathbf{q}_{b_i b_k}(\mathbf{a}^{b_k} - \mathbf{b}_k^a) + \mathbf{q}_{b_i b_{k+1}}(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)) \\
 \alpha_{b_i b_{k+1}} &= \alpha_{b_i b_k} + \beta_{b_i b_k}\delta t + \frac{1}{2}\mathbf{a}\delta t^2 \\
 \beta_{b_i b_{k+1}} &= \beta_{b_i b_k} + \mathbf{a}\delta t \\
 \mathbf{b}_{k+1}^a &= \mathbf{b}_k^a + \mathbf{n}_{b_k^a}\delta t \\
 \mathbf{b}_{k+1}^g &= \mathbf{b}_k^g + \mathbf{n}_{b_k^g}\delta t
 \end{aligned} \tag{34}$$

预积分量的方差

疑问

一个 IMU 数据作为测量值的噪声方差我们能够标定。现在，一段时间内多个 IMU 数据积分形成的预积分量的方差呢？

Covariance Propagation

已知一个变量 $y = \mathbf{Ax}$, $\mathbf{x} \in \mathcal{N}(0, \Sigma_x)$, 则有 $\Sigma_y = \mathbf{A}\Sigma_x\mathbf{A}^\top$

$$\begin{aligned}\Sigma_y &= E((\mathbf{Ax})(\mathbf{Ax})^\top) \\ &= E(\mathbf{Axx}^\top\mathbf{A}^\top) \\ &= \mathbf{A}\Sigma_x\mathbf{A}^\top\end{aligned}$$

所以，要推导预积分量的协方差，我们需要知道 imu 噪声和预积分量之间的线性递推关系。

假设已知了相邻时刻误差的线性传递方程：

$$\boldsymbol{\eta}_{ik} = \mathbf{F}_{k-1}\boldsymbol{\eta}_{ik-1} + \mathbf{G}_{k-1}\mathbf{n}_{k-1} \quad (35)$$

比如：状态量误差为 $\boldsymbol{\eta}_{ik} = [\delta\boldsymbol{\theta}_{ik}, \delta\mathbf{v}_{ik}, \delta\mathbf{p}_{ik}]$ ，测量噪声为 $\mathbf{n}_k = [\mathbf{n}_k^g, \mathbf{n}_k^a]$ 。

误差的传递由两部分组成：当前时刻的误差传递给下一时刻，当前时刻测量噪声传递给下一时刻。

一个有趣的例子

综艺节目中常有传递信息的节目，前一个人根据上一个人的信息 + 自己的理解（测量）传递给下一个人，导致这个信息越传越错。

协方差矩阵可以通过递推计算得到：

$$\boldsymbol{\Sigma}_{ik} = \mathbf{F}_{k-1}\boldsymbol{\Sigma}_{ik-1}\mathbf{F}_{k-1}^\top + \mathbf{G}_{k-1}\boldsymbol{\Sigma}_n\mathbf{G}_{k-1}^\top \quad (36)$$

其中， $\boldsymbol{\Sigma}_n$ 是测量噪声的协方差矩阵，方差从 i 时刻开始进行递推， $\boldsymbol{\Sigma}_{ii} = \mathbf{0}$ 。

状态误差线性递推公式的推导

简介

通常对于状态量之间的递推关系是非线性的方程如

$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$, 其中状态量为 \mathbf{x} , \mathbf{u} 为系统的输入量。

我们可以用两种方法来推导状态误差传递的线性递推关系：

- 一种是基于一阶泰勒展开的误差递推方程。
- 一种是基于误差随时间变化的递推方程。

基于一阶泰勒展开的误差递推方程

令状态量为 $\mathbf{x} = \hat{\mathbf{x}} + \delta\mathbf{x}$, 其中, 真值为 $\hat{\mathbf{x}}$, 误差为 $\delta\mathbf{x}$ 。另外, 输入量 \mathbf{u} 的噪声为 \mathbf{n} 。

基于泰勒展开的误差传递 (应用于 EKF 的协方差预测)

非线性系统 $\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$ 的状态误差的线性递推关系如下:

$$\delta\mathbf{x}_k = \mathbf{F}\delta\mathbf{x}_{k-1} + \mathbf{G}\mathbf{n}_{k-1} \quad (37)$$

其中, \mathbf{F} 是状态量 \mathbf{x}_k 对状态量 \mathbf{x}_{k-1} 的雅克比矩阵, \mathbf{G} 是状态量 \mathbf{x}_k 对输入量 \mathbf{u}_{k-1} 的雅克比矩阵。

证明: 对非线性状态方程进行一阶泰勒展开有:

$$\begin{aligned} \mathbf{x}_k &= f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \\ \hat{\mathbf{x}}_k + \delta\mathbf{x}_k &= f(\hat{\mathbf{x}}_{k-1} + \delta\mathbf{x}_{k-1}, \hat{\mathbf{u}}_{k-1} + \mathbf{n}_{k-1}) \\ \underline{\hat{\mathbf{x}}_k} + \delta\mathbf{x}_k &= \underline{f(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{u}}_{k-1})} + \mathbf{F}\delta\mathbf{x}_{k-1} + \mathbf{G}\mathbf{n}_{k-1} \end{aligned} \quad (38)$$

基于误差随时间变化的递推方程

基于误差随时间变化的递推方程

如果我们能够推导状态误差随时间变化的导数关系，比如：

$$\dot{\delta \mathbf{x}} = \mathbf{A}\delta \mathbf{x} + \mathbf{B}\mathbf{n} \quad (39)$$

则误差状态的传递方程为：

$$\begin{aligned} \delta \mathbf{x}_k &= \delta \mathbf{x}_{k-1} + \dot{\delta \mathbf{x}}_{k-1} \Delta t \\ \rightarrow \delta \mathbf{x}_k &= (\mathbf{I} + \mathbf{A}\Delta t)\delta \mathbf{x}_{k-1} + \mathbf{B}\Delta t \mathbf{n}_{k-1} \end{aligned} \quad (40)$$

两方法对比

这两种推导方式的可以看出有：

$$\mathbf{F} = \mathbf{I} + \mathbf{A}\Delta t, \quad \mathbf{G} = \mathbf{B}\Delta t \quad (41)$$

基于误差随时间变化的递推方程

第一种方法不是很好么，为什么会想着去弄误差随时间的变化呢？

这是因为 VIO 系统中已经知道了状态的导数和状态之间的转移矩阵。
如：我们已经知道速度和状态量之间的关系：

$$\dot{\mathbf{v}} = \mathbf{R}\mathbf{a}^b + \mathbf{g} \quad (42)$$

那我们就可以推导速度的误差和状态误差之间的关系，再每一项上都加上各自的误差就有： 这个可以认为是分治思想：将所有可能存在误差的状态量分离出来，那么他们形成的组合即是误差传递形式

$$\begin{aligned} \dot{\mathbf{v}} + \delta\dot{\mathbf{v}} &= \mathbf{R}(\mathbf{I} + [\delta\boldsymbol{\theta}]_{\times}) \left(\mathbf{a}^b + \delta\mathbf{a}^b \right) + \mathbf{g} + \delta\mathbf{g} \\ \delta\dot{\mathbf{v}} &= \mathbf{R}\delta\mathbf{a}^b + \mathbf{R}[\delta\boldsymbol{\theta}]_{\times} \left(\mathbf{a}^b + \delta\mathbf{a}^b \right) + \delta\mathbf{g} \\ \delta\dot{\mathbf{v}} &= \mathbf{R}\delta\mathbf{a}^b - \mathbf{R} \left[\mathbf{a}^b \right]_{\times} \delta\boldsymbol{\theta} + \delta\mathbf{g} \end{aligned} \quad (43)$$

由此就能以此类推，轻易写出整个 A 和 B 其他方程了。

预积分的误差递推公式推导

首先回顾预积分的误差递推公式。注意：这里白噪声用符号 $+$ ，表示噪声影响状态量。因为白噪声的值无法像 bias 一样估计，所以这里没办法减去白噪声：

$$\omega = \frac{1}{2}((\bar{\omega}^{b_k} + \mathbf{n}_k^g - \mathbf{b}_k^g) + (\bar{\omega}^{b_{k+1}} + \mathbf{n}_{k+1}^g - \mathbf{b}_k^g))$$

$$\mathbf{q}_{b_i b_{k+1}} = \mathbf{q}_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix}$$

$$\mathbf{a} = \frac{1}{2}(\mathbf{q}_{b_i b_k}(\bar{\mathbf{a}}^{b_k} + \mathbf{n}_k^a - \mathbf{b}_k^a) + \mathbf{q}_{b_i b_{k+1}}(\bar{\mathbf{a}}^{b_{k+1}} + \mathbf{n}_{k+1}^a - \mathbf{b}_k^a))$$

$$\alpha_{b_i b_{k+1}} = \alpha_{b_i b_k} + \beta_{b_i b_k} \delta t + \frac{1}{2} \mathbf{a} \delta t^2$$

$$\beta_{b_i b_{k+1}} = \beta_{b_i b_k} + \mathbf{a} \delta t$$

$$\mathbf{b}_{k+1}^a = \mathbf{b}_k^a + \mathbf{n}_{b_k^a} \delta t$$

$$\mathbf{b}_{k+1}^g = \mathbf{b}_k^g + \mathbf{n}_{b_k^g} \delta t$$

确定误差传递的状态量，噪声量，然后开始构建传递方程。

预积分误差传递的形式

用前面一阶泰勒展开的推导方式，我们希望能推导出误差的递推公式：

$$\begin{bmatrix} \delta \alpha_{b_{k+1}} \\ \delta \theta_{b_{k+1}} \\ \delta \beta_{b_{k+1}} \\ \delta \mathbf{b}_{k+1}^a \\ \delta \mathbf{b}_{k+1}^g \end{bmatrix} = \mathbf{F} \begin{bmatrix} \delta \alpha_{b_k} \\ \delta \theta_{b_k} \\ \delta \beta_{b_k} \\ \delta \mathbf{b}_k^a \\ \delta \mathbf{b}_k^g \end{bmatrix} + \mathbf{G} \begin{bmatrix} \mathbf{n}_k^a \\ \mathbf{n}_k^g \\ \mathbf{n}_{k+1}^a \\ \mathbf{n}_{k+1}^g \\ \mathbf{n}_{b_k^a} \\ \mathbf{n}_{b_k^g} \end{bmatrix} \quad (44)$$

\mathbf{F}, \mathbf{G} 为两个时刻间的协方差传递矩阵， $\delta(\cdot)$ 表示各时刻的误差。

这里我们直接给出 \mathbf{F} , \mathbf{G} 的最终形式, 后面会对部分项进行详细推导:

$$\mathbf{F} = \begin{bmatrix} \mathbf{I} & \mathbf{f}_{12} & \mathbf{I}\delta t & -\frac{1}{4}(\mathbf{q}_{b_i b_k} + \mathbf{q}_{b_i b_{k+1}})\delta t^2 & \mathbf{f}_{15} \\ \mathbf{0} & \mathbf{I} - [\boldsymbol{\omega}]_{\times}\delta t & \mathbf{0} & \mathbf{0} & -\mathbf{I}\delta t \\ \mathbf{0} & \mathbf{f}_{32} & \mathbf{I} & -\frac{1}{2}(\mathbf{q}_{b_i b_k} + \mathbf{q}_{b_i b_{k+1}})\delta t & \mathbf{f}_{35} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \quad (45)$$

$$\mathbf{G} = \begin{bmatrix} \frac{1}{4}\mathbf{q}_{b_i b_k}\delta t^2 & \mathbf{g}_{12} & \frac{1}{4}\mathbf{q}_{b_i b_{k+1}}\delta t^2 & \mathbf{g}_{14} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\mathbf{I}\delta t & \mathbf{0} & \frac{1}{2}\mathbf{I}\delta t & \mathbf{0} & \mathbf{0} \\ \frac{1}{2}\mathbf{q}_{b_i b_k}\delta t & \mathbf{g}_{32} & \frac{1}{2}\mathbf{q}_{b_i b_{k+1}}\delta t & \mathbf{g}_{34} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\delta t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\delta t \end{bmatrix} \quad (46)$$

其中的系数为：

$$\begin{aligned}
 \mathbf{f}_{12} &= \frac{\partial \delta \alpha_{b_{k+1}}}{\partial \delta \theta_{b_k}} = -\frac{1}{4}(\mathbf{R}_{b_i b_k}[\mathbf{a}^{b_k} - \mathbf{b}_k^a] \times \delta t^2 + \mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)] \times (\mathbf{I} - [\omega] \times \delta t) \delta t^2) \\
 \mathbf{f}_{32} &= \frac{\partial \delta \beta_{b_{k+1}}}{\partial \delta \theta_{b_k}} = -\frac{1}{2}(\mathbf{R}_{b_i b_k}[\mathbf{a}^{b_k} - \mathbf{b}_k^a] \times \delta t + \mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)] \times (\mathbf{I} - [\omega] \times \delta t) \delta t) \\
 \mathbf{f}_{15} &= \frac{\partial \delta \alpha_{b_{k+1}}}{\partial \delta \mathbf{b}_k^g} = -\frac{1}{4}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)] \times \delta t^2)(-\delta t) \\
 \mathbf{f}_{35} &= \frac{\partial \delta \beta_{b_{k+1}}}{\partial \delta \mathbf{b}_k^g} = -\frac{1}{2}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)] \times \delta t)(-\delta t) \\
 \mathbf{g}_{12} &= \frac{\partial \delta \alpha_{b_{k+1}}}{\partial \mathbf{n}_k^g} = \mathbf{g}_{14} = \frac{\partial \delta \alpha_{b_{k+1}}}{\partial \mathbf{n}_{k+1}^g} = -\frac{1}{4}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)] \times \delta t^2)(\frac{1}{2}\delta t) \\
 \mathbf{g}_{32} &= \frac{\partial \delta \beta_{b_{k+1}}}{\partial \mathbf{n}_k^g} = \mathbf{g}_{34} = \frac{\partial \delta \beta_{b_{k+1}}}{\partial \mathbf{n}_{k+1}^g} = -\frac{1}{2}(\mathbf{R}_{b_i b_{k+1}}[(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)] \times \delta t)(\frac{1}{2}\delta t)
 \end{aligned} \tag{47}$$

注意，以上形式是为了跟代码一致，所以并没有进一步约简。

雅克比矩阵 F, G 的推导

公式简化约定

考虑到公式的编辑篇幅，为了对一些求导公式进行简化，这里做一些简单的约定，比如求导公式：

$$\frac{\partial \mathbf{x}_a}{\partial \delta \boldsymbol{\theta}} = \lim_{\delta \boldsymbol{\theta} \rightarrow 0} \frac{\mathbf{R}_{ab} \exp([\delta \boldsymbol{\theta}]_{\times}) \mathbf{x}_b - \mathbf{R}_{ab} \mathbf{x}_b}{\delta \boldsymbol{\theta}}$$

后续直接简写为

$$\frac{\partial \mathbf{x}_a}{\partial \delta \boldsymbol{\theta}} = \frac{\partial \mathbf{R}_{ab} \exp([\delta \boldsymbol{\theta}]_{\times}) \mathbf{x}_b}{\partial \delta \boldsymbol{\theta}}$$

雅克比矩阵 F 的推导

β 对各状态量的雅克比推导，即 F 第三行

速度预积分量 β 的递推计算形式：

$$\begin{aligned}\beta_{b_i b_{k+1}} &= \beta_{b_i b_k} + \mathbf{a} \delta t \\ &= \beta_{b_i b_k} + \frac{1}{2}(\mathbf{q}_{b_i b_k}(\mathbf{a}^{b_k} - \mathbf{b}_k^a) + \mathbf{q}_{b_i b_{k+1}}(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a))\delta t\end{aligned}\quad (48)$$

\mathbf{f}_{33} : 对上一时刻速度预积分量的 Jacobian

$$\hat{\beta}_{b_i b_{k+1}} + \delta \beta_{b_{k+1}} = \hat{\beta}_{b_i b_k} + \delta \beta_{b_k} + \mathbf{a} \delta t \longrightarrow \mathbf{f}_{33} = \frac{\partial \delta \beta_{b_{k+1}}}{\partial \delta \beta_{b_k}} = \mathbf{I}_{3 \times 3} \quad (49)$$

\mathbf{f}_{32} : 对角度预积分量的 Jacobian

首先将公式(48)写成如下形式：

$$\beta_{b_i b_{k+1}} = \beta_{b_i b_k} + \frac{1}{2}(\mathbf{q}_{b_i b_k}(\mathbf{a}^{b_k} - \mathbf{b}_k^a) + \mathbf{q}_{b_i b_k} \otimes \left[\frac{1}{2} \boldsymbol{\omega} \delta t \right] (\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a))\delta t \quad (50)$$

f_{32} : 对角度预积分量的 Jacobian

那么, 从公式 (48) 易知, 速度的预积分量对角度预积分量误差 $\delta\beta_{b_k}$ 的 Jacobian 只跟加速度项有关:

$$\begin{aligned}
 \mathbf{a}\delta t &= \frac{1}{2}\hat{\mathbf{q}}_{b_i b_k} \otimes \left[\frac{1}{2}\delta\boldsymbol{\theta}_{b_k} \right] (\mathbf{a}^{b_k} - \mathbf{b}_k^a) \delta t \\
 &\quad + \frac{1}{2}\hat{\mathbf{q}}_{b_i b_k} \otimes \left[\frac{1}{2}\delta\boldsymbol{\theta}_{b_k} \right] \otimes \left[\frac{1}{2}\boldsymbol{\omega}\delta t \right] (\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t \\
 &= \underbrace{\frac{1}{2}\mathbf{R}_{b_i b_k} \exp([\delta\boldsymbol{\theta}_{b_k}]_{\times})(\mathbf{a}^{b_k} - \mathbf{b}_k^a)\delta t}_{\text{Part 1}} \\
 &\quad + \underbrace{\frac{1}{2}\mathbf{R}_{b_i b_k} \exp([\delta\boldsymbol{\theta}_{b_k}]_{\times}) \exp([\boldsymbol{\omega}\delta t]_{\times})(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)\delta t}_{\text{Part 2}}
 \end{aligned} \tag{51}$$

f_{32} : 对角度预积分量误差的 Jacobian

Part 1 对应的的雅克比为:

$$\begin{aligned}
 \frac{\partial \mathbf{R}_{b_i b_k} \exp([\delta \boldsymbol{\theta}_{b_k}]_{\times})(\mathbf{a}^{b_k} - \mathbf{b}_k^a) \delta t}{\partial \delta \boldsymbol{\theta}_{b_k}} &= \frac{\partial \mathbf{R}_{b_i b_k} (\mathbf{I} + [\delta \boldsymbol{\theta}_{b_k}]_{\times})(\mathbf{a}^{b_k} - \mathbf{b}_k^a) \delta t}{\partial \delta \boldsymbol{\theta}_{b_k}} \\
 &= \frac{\partial - \mathbf{R}_{b_i b_k} [(\mathbf{a}^{b_k} - \mathbf{b}_k^a) \delta t]_{\times} \delta \boldsymbol{\theta}_{b_k}}{\partial \delta \boldsymbol{\theta}_{b_k}} \\
 &= -\mathbf{R}_{b_i b_k} [(\mathbf{a}^{b_k} - \mathbf{b}_k^a) \delta t]_{\times}
 \end{aligned} \tag{52}$$

Part 2 对应的的雅克比为:

$$\begin{aligned}
 & \frac{\partial \mathbf{R}_{b_i b_k} \exp([\delta \boldsymbol{\theta}_{b_k}]_{\times}) \exp([\boldsymbol{\omega} \delta t]_{\times})(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t}{\partial \delta \boldsymbol{\theta}_{b_k}} \\
 &= \frac{\partial \mathbf{R}_{b_i b_k} (\mathbf{I} + [\delta \boldsymbol{\theta}_{b_k}]_{\times}) \exp([\boldsymbol{\omega} \delta t]_{\times})(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t}{\partial \delta \boldsymbol{\theta}_{b_k}} \\
 &= - \frac{\partial \mathbf{R}_{b_i b_k} [\exp([\boldsymbol{\omega} \delta t]_{\times})(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t]_{\times} \delta \boldsymbol{\theta}_{b_k}}{\partial \delta \boldsymbol{\theta}_{b_k}} \\
 &= - \frac{\partial \mathbf{R}_{b_i b_k} \exp([\boldsymbol{\omega} \delta t]_{\times}) [(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t]_{\times} \exp([- \boldsymbol{\omega} \delta t]_{\times}) \delta \boldsymbol{\theta}_{b_k}}{\partial \delta \boldsymbol{\theta}_{b_k}} \\
 &= - \mathbf{R}_{b_i b_{k+1}} [(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t]_{\times} \exp([- \boldsymbol{\omega} \delta t]_{\times}) \\
 &\approx - \mathbf{R}_{b_i b_{k+1}} [(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t]_{\times} (\mathbf{I} - [\boldsymbol{\omega} \delta t]_{\times})
 \end{aligned} \tag{53}$$

将上面两部分综合起来就能得到

$$\mathbf{f}_{32} = \frac{\partial \delta \beta_{b_{k+1}}}{\partial \delta \boldsymbol{\theta}_{b_k}} = -\frac{1}{2} (\mathbf{R}_{b_i b_k} [\mathbf{a}^{b_k} - \mathbf{b}_k^a]_{\times} \delta t + \mathbf{R}_{b_i b_{k+1}} [(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)]_{\times} (\mathbf{I} - [\boldsymbol{\omega}]_{\times} \delta t) \delta t)$$

f₃₅: 速度预积分量对 k 时刻角速度 bias 的 Jacobian

递推公式如下:

$$\omega = \frac{1}{2}((\omega^{b_k} - \mathbf{b}_k^g) + (\omega^{b_{k+1}} - \mathbf{b}_k^g)) = \frac{1}{2}(\omega^{b_k} + \omega^{b_{k+1}}) - \mathbf{b}_k^g$$

$$\beta_{b_i b_{k+1}} = \beta_{b_i b_k} + \underbrace{\frac{1}{2}(\mathbf{q}_{b_i b_k}(\mathbf{a}^{b_k} - \mathbf{b}_k^a) + \mathbf{q}_{b_i b_k} \otimes \left[\begin{array}{c} 1 \\ \frac{1}{2}\omega\delta t \end{array} \right] (\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a))\delta t}_{\text{red part}}$$

只有红色公式部分和角速度 bias 有关系, 因此雅克比的推导只考虑红色公式部分。

$$\begin{aligned}
\mathbf{f}_{35} &= \frac{\partial \delta \beta_{b_{k+1}}}{\partial \delta \mathbf{b}_k^g} \\
&= \frac{\partial \frac{1}{2} \mathbf{q}_{b_i b_k} \otimes \left[\frac{1}{2} (\boldsymbol{\omega} - \delta \mathbf{b}_k^g) \delta t \right] (\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t}{\partial \delta \mathbf{b}_k^g} \\
&= \frac{1}{2} \frac{\partial \mathbf{R}_{b_i b_k} \exp \left([(\boldsymbol{\omega} - \delta \mathbf{b}_k^g) \delta t]_{\times} \right) (\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t}{\partial \delta \mathbf{b}_k^g} \\
&= \frac{1}{2} \frac{\partial \mathbf{R}_{b_i b_k} \exp \left([\boldsymbol{\omega} \delta t]_{\times} \right) \exp \left([-J_r (\boldsymbol{\omega} \delta t) \delta \mathbf{b}_k^g \delta t]_{\times} \right) (\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t}{\partial \delta \mathbf{b}_k^g} \\
&= \frac{1}{2} \frac{\partial - \mathbf{R}_{b_i b_{k+1}} \left([(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a) \delta t]_{\times} \right) (-J_r (\boldsymbol{\omega} \delta t) \delta \mathbf{b}_k^g \delta t)}{\partial \delta \mathbf{b}_k^g} \\
&= -\frac{1}{2} (\mathbf{R}_{b_i b_{k+1}} [(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)]_{\times} \delta t) (-J_r (\boldsymbol{\omega} \delta t) \delta t)
\end{aligned} \tag{54}$$

其中利用公式： $\exp \left([\phi + \delta \phi]_{\times} \right) \approx \exp([\phi]_{\times}) \exp \left([J_r(\phi) \delta \phi]_{\times} \right)$, $J_r(\phi)$ 称之为 SO3 的右雅克比。当 ϕ 非常小时, $J_r(\phi) \approx \mathbf{I}$.

旋转预积分量的 Jacobian，即 F 第二行

旋转预积分的递推公式为：

$$\begin{aligned}
 \omega &= \frac{1}{2}((\omega^{b_k} - \mathbf{b}_k^g) + (\omega^{b_{k+1}} - \mathbf{b}_k^g)) \\
 \mathbf{q}_{b_i b_{k+1}} &= \mathbf{q}_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\omega\delta t \end{bmatrix} \\
 &= \mathbf{q}_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(\frac{1}{2}(\omega^{b_k} + \omega^{b_{k+1}}) - \mathbf{b}_k^g)\delta t \end{bmatrix}
 \end{aligned} \tag{55}$$

f₂₂: 前一时刻的旋转误差 $\delta\theta_{b_k}$ 如何影响当前旋转误差 $\delta\theta_{b_{k+1}}$

假设两个时刻的真值为 $\mathbf{q}_{b_i b_{k+1}}$, $\mathbf{q}_{b_i b_k}$, 两个时刻间的增量真值为 $\mathbf{q}_{b_k b_{k+1}}$ 。推导过程只考虑一个变量, 即旋转误差 $\delta\theta_{b_k}$ 的影响, 而不考虑测量值角速度 bias 误差影响。可以假设 $\mathbf{q}_{b_k b_{k+1}} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\boldsymbol{\omega}\delta t \end{bmatrix}$ 。

另外, 三元组四元数相乘有如下性质:

$$\mathbf{q} \otimes \mathbf{p} \otimes \mathbf{q}^* = [\mathbf{q}]_L [\mathbf{q}]_R^\top \mathbf{p} = \begin{bmatrix} p_w \\ \mathbf{R}\mathbf{p}_v \end{bmatrix} \quad (56)$$

其中 \mathbf{R} 是和 \mathbf{q} 对应的旋转矩阵, p_w 为 \mathbf{p} 的实部, \mathbf{p}_v 为 \mathbf{p} 的虚部。

下面开始详细推导：

$$\begin{aligned}
 \mathbf{q}_{b_i b_{k+1}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_{k+1}} \end{bmatrix} &= \mathbf{q}_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_k} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \boldsymbol{\omega} \delta t \end{bmatrix} \\
 \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_{k+1}} \end{bmatrix} &= \mathbf{q}_{b_i b_{k+1}}^* \otimes \mathbf{q}_{b_i b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_k} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \boldsymbol{\omega} \delta t \end{bmatrix} \\
 &= \mathbf{q}_{b_{k+1} b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_k} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \boldsymbol{\omega} \delta t \end{bmatrix} \\
 &\approx \begin{bmatrix} 1 \\ -\frac{1}{2} \boldsymbol{\omega} \delta t \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_k} \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \boldsymbol{\omega} \delta t \end{bmatrix} \\
 &= \begin{bmatrix} 1 \\ \frac{1}{2} \mathbf{R} \delta \boldsymbol{\theta}_{b_k} \end{bmatrix}
 \end{aligned} \tag{57}$$

注意：上面推导过程，也可以用李代数的右扰动 $\mathbf{R} \exp([\delta \boldsymbol{\theta}_{k+1}]_{\times})$

雅克比 f_{22} 的推导

只考虑公式(57) 中的虚部:

$$\begin{aligned}\delta\theta_{b_{k+1}} &= \mathbf{R}\delta\theta_{b_k} \\ &= \exp([- \boldsymbol{\omega}\delta t]_{\times})\delta\theta_{b_k} \\ &\approx (\mathbf{I} - [\boldsymbol{\omega}\delta t]_{\times})\delta\theta_{b_k}\end{aligned}\tag{58}$$

那么, 第 $k+1$ 时刻的旋转预积分的误差相对于第 k 时刻的 Jacobian 为:

$$\mathbf{f}_{22} = \frac{\partial\delta\theta_{b_{k+1}}}{\partial\delta\theta_{b_k}} = \mathbf{I} - [\boldsymbol{\omega}\delta t]_{\times}\tag{59}$$

渔已授完，F, G 中的其他鱼靠大家去捞了...

Section 4

残差 Jacobian 的推导

视觉重投影残差的 Jacobian

视觉残差为：

$$\mathbf{r}_c = \begin{bmatrix} \frac{x_{c_j}}{z_{c_j}} - u_{c_j} \\ \frac{y_{c_j}}{z_{c_j}} - v_{c_j} \\ z_{c_j} \end{bmatrix}$$

对于第 i 帧中的特征点，它投影到第 j 帧相机坐标系下的值为：

$$\begin{bmatrix} x_{c_j} \\ y_{c_j} \\ z_{c_j} \\ 1 \end{bmatrix} = \mathbf{T}_{bc}^{-1} \mathbf{T}_{wb_j}^{-1} \mathbf{T}_{wb_i} \mathbf{T}_{bc} \begin{bmatrix} \frac{1}{\lambda} u_{c_i} \\ \frac{1}{\lambda} v_{c_i} \\ \frac{1}{\lambda} \\ 1 \end{bmatrix}$$

拆成三维坐标形式为：

$$\begin{aligned} \mathbf{f}_{c_j} = \begin{bmatrix} x_{c_j} \\ y_{c_j} \\ z_{c_j} \end{bmatrix} &= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \frac{1}{\lambda} \begin{bmatrix} u_{c_i} \\ v_{c_i} \\ 1 \end{bmatrix} \\ &+ \mathbf{R}_{bc}^\top (\mathbf{R}_{wb_j}^\top ((\mathbf{R}_{wb_i} \mathbf{p}_{bc} + \mathbf{p}_{wb_i}) - \mathbf{p}_{wb_j}) - \mathbf{p}_{bc}) \end{aligned} \quad (60)$$

再推导各类 Jacobian 之前，为了简化公式，先定义如下变量：

$$\begin{aligned}\mathbf{f}_{b_i} &= \mathbf{R}_{bc}\mathbf{f}_{c_i} + \mathbf{p}_{bc} \\ \mathbf{f}_w &= \mathbf{R}_{wb_i}\mathbf{f}_{b_i} + \mathbf{p}_{wb_i} \\ \mathbf{f}_{b_j} &= \mathbf{R}_{wb_j}^\top(\mathbf{f}_w - \mathbf{p}_{wb_j})\end{aligned}\quad (61)$$

Jacobian 为视觉误差对两个时刻的状态量，外参，以及逆深度求导：

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{r}_c}{\partial \begin{bmatrix} \delta \mathbf{p}_{b_i b'_i} \\ \delta \boldsymbol{\theta}_{b_i b'_i} \end{bmatrix}} & \frac{\partial \mathbf{r}_c}{\partial \begin{bmatrix} \delta \mathbf{p}_{b_j b'_j} \\ \delta \boldsymbol{\theta}_{b_j b'_j} \end{bmatrix}} & \frac{\partial \mathbf{r}_c}{\partial \begin{bmatrix} \delta \mathbf{p}_{cc'} \\ \delta \boldsymbol{\theta}_{cc'} \end{bmatrix}} & \frac{\partial \mathbf{r}_c}{\partial \delta \lambda} \end{bmatrix} \quad (62)$$

根据链式法则, Jacobian 的计算可以分两步走, 第一步误差对 \mathbf{f}_{c_j} 求导:

$$\frac{\partial \mathbf{r}_c}{\partial \mathbf{f}_{c_j}} = \begin{bmatrix} \frac{1}{z_{c_j}} & 0 & -\frac{x_{c_j}}{z_{c_j}^2} \\ 0 & \frac{1}{z_{c_j}} & -\frac{y_{c_j}}{z_{c_j}^2} \end{bmatrix} \quad (63)$$

第二步 \mathbf{f}_{c_j} 对各状态量求导:

1. 对 i 时刻的状态量求导

a. 对 i 时刻位移求导, 可直接写出如下:

$$\frac{\partial \mathbf{f}_{c_j}}{\partial \delta \mathbf{p}_{b_i b'_i}} = \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \quad (64)$$

b. 对 i 时刻角度增量求导

$$\mathbf{f}_{c_j} = \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{f}_{c_i} + \mathbf{R}_{bc}^\top (\mathbf{R}_{wb_j}^\top ((\mathbf{R}_{wb_i} \mathbf{p}_{bc} + \mathbf{p}_{wb_i}) - \mathbf{p}_{wb_j}) - \mathbf{p}_{bc}) \quad (65)$$

上面公式和 i 时刻角度相关的量并不多，下面为了简化，直接丢弃了不相关的部分

$$\begin{aligned}
 \mathbf{f}_{c_j} &= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{f}_{c_i} + \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{p}_{bc} + (\dots) \\
 &= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} (\mathbf{R}_{bc} \mathbf{f}_{c_i} + \mathbf{p}_{bc}) + (\dots) \\
 &= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{f}_{b_i} + (\dots)
 \end{aligned} \tag{66}$$

Jacobian 为

$$\begin{aligned}
 \frac{\partial \mathbf{f}_{c_j}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} &= \frac{\partial \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} (\mathbf{I} + [\delta \boldsymbol{\theta}_{b_i b'_i}]_\times) \mathbf{f}_{b_i}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
 &= -\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} [\mathbf{f}_{b_i}]_\times
 \end{aligned} \tag{67}$$

2. 对 j 时刻的状态量求导

a. 对位移求导：

$$\frac{\partial \mathbf{f}_{c_j}}{\partial \delta \mathbf{p}_{b_j b'_j}} = -\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \quad (68)$$

b. 对角度增量求导，同上面的操作，也简化一下公式

$$\begin{aligned} \mathbf{f}_{c_j} &= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{f}_{c_i} + \mathbf{R}_{bc}^\top (\mathbf{R}_{wb_j}^\top ((\mathbf{R}_{wb_i} \mathbf{p}_{bc} + \mathbf{p}_{wb_i}) - \mathbf{p}_{wb_j}) - \mathbf{p}_{bc}) \\ &= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top (\mathbf{R}_{wb_i} (\mathbf{R}_{bc} \mathbf{f}_{c_i} + \mathbf{p}_{bc}) + \mathbf{p}_{wb_i} - \mathbf{p}_{wb_j}) + (\dots) \\ &= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top (\mathbf{f}_w - \mathbf{p}_{wb_j}) + (\dots) \end{aligned} \quad (69)$$

Jacobian 为

$$\begin{aligned} \frac{\partial \mathbf{f}_{c_j}}{\partial \delta \boldsymbol{\theta}_{b_j b'_j}} &= \frac{\partial \mathbf{R}_{bc}^\top (\mathbf{I} - [\delta \boldsymbol{\theta}_{b_j b'_j}]_\times) \mathbf{R}_{wb_j}^\top (\mathbf{f}_w - \mathbf{p}_{wb_j})}{\partial \delta \boldsymbol{\theta}_{b_j b'_j}} \\ &= \frac{\partial \mathbf{R}_{bc}^\top (\mathbf{I} - [\delta \boldsymbol{\theta}_{b_j b'_j}]_\times) \mathbf{f}_{b_j}}{\partial \delta \boldsymbol{\theta}_{b_j b'_j}} \\ &= \mathbf{R}_{bc}^\top [\mathbf{f}_{b_j}]_\times \end{aligned} \quad (70)$$

3. 对 imu 和相机之间的外参求导

a. 对位移求导

$$\frac{\partial \mathbf{f}_{c_j}}{\partial \delta \mathbf{p}_{cc'}} = \mathbf{R}_{bc}^\top (\mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} - \mathbf{I}_{3 \times 3}) \quad (71)$$

b. 对角度增量求导，由于 \mathbf{f}_{c_j} 都和 \mathbf{R}_{bc} 有关，并且比较复杂，所以这次分两部分求导

$$\begin{aligned} \mathbf{f}_{c_j} &= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{f}_{c_i} + \mathbf{R}_{bc}^\top (\mathbf{R}_{wb_j}^\top ((\mathbf{R}_{wb_i} \mathbf{p}_{bc} + \mathbf{p}_{wb_i}) - \mathbf{p}_{wb_j}) - \mathbf{p}_{bc}) \\ &= \mathbf{f}_{c_j}^1 + \mathbf{f}_{c_j}^2 \end{aligned} \quad (72)$$

第一部分 Jacobian 为

$$\frac{\partial \mathbf{f}_{c_j}^1}{\partial \delta \boldsymbol{\theta}_{cc'}} = \frac{\partial (\mathbf{I} - [\delta \boldsymbol{\theta}_{cc'}]_\times) \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} (\mathbf{I} + [\delta \boldsymbol{\theta}_{cc'}]_\times) \mathbf{f}_{c_i}}{\partial \delta \boldsymbol{\theta}_{cc'}} \quad (73)$$

分子可写成：

$$\begin{aligned} &\partial \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} [\delta \boldsymbol{\theta}_{cc'}]_\times \mathbf{f}_{c_i} - [\delta \boldsymbol{\theta}_{cc'}]_\times \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{f}_{c_i} \\ &+ o^2(\delta \boldsymbol{\theta}_{cc'}) + (\dots) \end{aligned}$$

那么，第一部分的 Jacobian 为：

$$\frac{\partial \mathbf{f}_{c_j}^1}{\partial \delta \boldsymbol{\theta}_{cc'}} = -\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} [\mathbf{f}_{c_i}]_{\times} + [\mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{f}_{c_i}]_{\times} \quad (74)$$

第二部分的 Jacobian 为：

$$\begin{aligned} \frac{\partial \mathbf{f}_{c_j}^2}{\partial \delta \boldsymbol{\theta}_{cc'}} &= \frac{(\mathbf{I} - [\delta \boldsymbol{\theta}_{cc'}]_{\times}) \mathbf{R}_{bc}^\top (\mathbf{R}_{wb_j}^\top ((\mathbf{R}_{wb_i} \mathbf{p}_{bc} + \mathbf{p}_{wb_i}) - \mathbf{p}_{wb_j}) - \mathbf{p}_{bc})}{\partial \delta \boldsymbol{\theta}_{cc'}} \\ &= [\mathbf{R}_{bc}^\top (\mathbf{R}_{wb_j}^\top ((\mathbf{R}_{wb_i} \mathbf{p}_{bc} + \mathbf{p}_{wb_i}) - \mathbf{p}_{wb_j}) - \mathbf{p}_{bc})]_{\times} \end{aligned} \quad (75)$$

两个 Jacobian 相加就是视觉误差对外参中的角度增量的最终结果。

3. 视觉误差对特征逆深度的求导

$$\begin{aligned}
 \frac{\partial \mathbf{f}_{c_j}}{\partial \delta \lambda} &= \frac{\partial \mathbf{f}_{c_j}}{\partial \mathbf{f}_{c_i}} \frac{\partial \mathbf{f}_{c_i}}{\partial \delta \lambda} \\
 &= \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \left(-\frac{1}{\lambda^2} \begin{bmatrix} u_{c_i} \\ v_{c_i} \\ 1 \end{bmatrix} \right) \\
 &= -\frac{1}{\lambda} \mathbf{R}_{bc}^\top \mathbf{R}_{wb_j}^\top \mathbf{R}_{wb_i} \mathbf{R}_{bc} \mathbf{f}_{c_i}
 \end{aligned} \tag{76}$$

IMU 误差相对于优化变量的 Jacobian

在求解非线性方程时，需要知道误差 \mathbf{e}_B 对两个关键帧 i, j 的状态量 $\mathbf{p}, \mathbf{q}, \mathbf{v}, \mathbf{b}^a, \mathbf{b}^g$ 的 Jacobian。

$$\mathbf{e}_B(x_i, x_j) = \begin{bmatrix} \mathbf{r}_p \\ \mathbf{r}_q \\ \mathbf{r}_v \\ \mathbf{r}_{ba} \\ \mathbf{r}_{bg} \end{bmatrix} = \begin{bmatrix} \mathbf{q}_{b_i w}(\mathbf{p}_{wb_j} - \mathbf{p}_{wb_i} - \mathbf{v}_i^w \Delta t + \frac{1}{2} \mathbf{g}^w \Delta t^2) - \alpha_{b_i b_j} \\ 2[\mathbf{q}_{b_j b_i} \otimes (\mathbf{q}_{b_i w} \otimes \mathbf{q}_{wb_j})]_{xyz} \\ \mathbf{q}_{b_i w}(\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t) - \beta_{b_i b_j} \\ \mathbf{b}_j^a - \mathbf{b}_i^a \\ \mathbf{b}_j^g - \mathbf{b}_i^g \end{bmatrix}_{15 \times 1}$$

对 i, j 时刻的状态量 $\mathbf{p}, \mathbf{q}, \mathbf{v}$ 求导还是比较直观的，直接对误差公式进行计算就行。但是对 i 时刻的 $\mathbf{b}_i^a, \mathbf{b}_i^g$ 求导就显得十分复杂，下面我们详细讨论。

因为 i 时刻的 bias 相关的预积分计算是通过迭代一步一步累计递推的，可以算但是太复杂。所以对于预积分量直接在 i 时刻的 bias 附近用一阶泰勒展开来近似，而不用真的去迭代计算。

$$\begin{aligned}
 \alpha_{b_i b_j} &= \alpha_{b_i b_j} + \mathbf{J}_{b_i^a}^\alpha \delta \mathbf{b}_i^a + \mathbf{J}_{b_i^g}^\alpha \delta \mathbf{b}_i^g \\
 \beta_{b_i b_j} &= \beta_{b_i b_j} + \mathbf{J}_{b_i^a}^\beta \delta \mathbf{b}_i^a + \mathbf{J}_{b_i^g}^\beta \delta \mathbf{b}_i^g \\
 \mathbf{q}_{b_i b_j} &= \mathbf{q}_{b_i b_j} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \mathbf{J}_{b_i^g}^q \delta \mathbf{b}_i^g \end{bmatrix}
 \end{aligned} \tag{77}$$

其中 $\mathbf{J}_{b_i^a}^\alpha = \frac{\partial \alpha_{b_i b_j}}{\partial \delta \mathbf{b}_i^a}$, $\mathbf{J}_{b_i^g}^\alpha = \frac{\partial \alpha_{b_i b_j}}{\partial \delta \mathbf{b}_i^g}$, $\mathbf{J}_{b_i^a}^\beta = \frac{\partial \beta_{b_i b_j}}{\partial \delta \mathbf{b}_i^a}$, $\mathbf{J}_{b_i^g}^\beta = \frac{\partial \beta_{b_i b_j}}{\partial \delta \mathbf{b}_i^g}$, $\mathbf{J}_{b_i^g}^q = \frac{\partial \mathbf{q}_{b_i b_j}}{\partial \delta \mathbf{b}_i^g}$

表示预积分量对 i 时刻的 bias 求导。

这些雅克比根据前面讨论的协方差传递公式，能一步步递推得到：

$$\mathbf{J}_{k+1} = \mathbf{F}_k \mathbf{J}_k \tag{78}$$

下面我们来讨论 IMU 误差相对于两帧的 PVQ 的 Jacobian:

由于 \mathbf{r}_p 和 \mathbf{r}_v 的误差形式很相近, 对各状态量求导的 Jacobian 形式也很相似, 所以这里只对 \mathbf{r}_v 的推导进行详细介绍。

(1) 对 i 时刻位移 Jacobian

$$\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{p}_{b_i b'_i}} = \mathbf{0} \quad (79)$$

(2) 对 i 时刻旋转 Jacobian

$$\begin{aligned} \frac{\partial \mathbf{r}_v}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} &= \frac{\partial (\mathbf{q}_{wb_i} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_i b'_i} \end{bmatrix})^{-1} (\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t)}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\ &= \frac{\partial (\mathbf{R}_{wb_i} \exp([\delta \boldsymbol{\theta}_{b_i b'_i}]_{\times}))^{-1} (\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t)}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\ &= \frac{\partial \exp([- \delta \boldsymbol{\theta}_{b_i b'_i}]_{\times}) \mathbf{R}_{b_i w} (\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t)}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \end{aligned} \quad (80)$$

上式可写为:

$$\begin{aligned}
 \frac{\partial \mathbf{r}_v}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} &= \frac{\partial (\mathbf{I} - [\delta \boldsymbol{\theta}_{b_i b'_i}]_{\times}) \mathbf{R}_{b_i w} (\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t)}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
 &= \frac{\partial - [\delta \boldsymbol{\theta}_{b_i b'_i}]_{\times} \mathbf{R}_{b_i w} (\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t)}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
 &= \frac{\partial [\mathbf{R}_{b_i w} (\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t)]_{\times} \delta \boldsymbol{\theta}_{b_i b'_i}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
 &= [\mathbf{R}_{b_i w} (\mathbf{v}_j^w - \mathbf{v}_i^w + \mathbf{g}^w \Delta t)]_{\times}
 \end{aligned} \tag{81}$$

(3) 对 i 时刻速度 Jacobian:

$$\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{v}_i^w} = -\mathbf{R}_{b_i w} \tag{82}$$

(4) 对 i 时刻的加速度 bias 的 Jacobian, 注意 bias 量只和预积分 β 有关:

$$\frac{\partial \mathbf{r}_v}{\partial \delta \mathbf{b}_i^a} = -\frac{\partial \beta_{b_i b_j}}{\partial \delta \mathbf{b}_i^a} = -\mathbf{J}_{b_i^a}^{\beta} \tag{83}$$

IMU 角度误差相对优化变量的 Jacobian

(1) 对 i 时刻姿态求导

$$\begin{aligned}
 \frac{\partial \mathbf{r}_q}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} &= \frac{\partial 2[\mathbf{q}_{b_j b_i} \otimes (\mathbf{q}_{b_i w} \otimes \mathbf{q}_{w b_j})]_{xyz}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
 &= \frac{\partial 2[\mathbf{q}_{b_i b_j}^* \otimes (\mathbf{q}_{w b_i} \otimes \left[\begin{smallmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_i b'_i} \end{smallmatrix} \right])^* \otimes \mathbf{q}_{w b_j}]_{xyz}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
 &= \frac{\partial - 2[(\mathbf{q}_{b_i b_j}^* \otimes (\mathbf{q}_{w b_i} \otimes \left[\begin{smallmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_i b'_i} \end{smallmatrix} \right])^* \otimes \mathbf{q}_{w b_j})^*]_{xyz}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
 &= \frac{\partial - 2[\mathbf{q}_{w b_j}^* \otimes (\mathbf{q}_{w b_i} \otimes \left[\begin{smallmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_i b'_i} \end{smallmatrix} \right]) \otimes \mathbf{q}_{b_i b_j}]_{xyz}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}}
 \end{aligned} \tag{84}$$

把这部分看作整体，那么则有 $\mathbf{q} = -\mathbf{q}^*$

上式可化简为：

$$\begin{aligned}
 \frac{\partial \mathbf{r}_q}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} &= -2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \frac{\mathbf{q}_{wb_j}^* \otimes (\mathbf{q}_{wb_i} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_i b'_i} \end{bmatrix}) \otimes \mathbf{q}_{b_i b_j}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
 &= -2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \frac{\partial [\mathbf{q}_{wb_j}^* \otimes \mathbf{q}_{wb_i}]_L [\mathbf{q}_{b_i b_j}]_R \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_i b'_i} \end{bmatrix}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
 &= -2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} [\mathbf{q}_{wb_j}^* \otimes \mathbf{q}_{wb_i}]_L [\mathbf{q}_{b_i b_j}]_R \begin{bmatrix} \mathbf{0} \\ \frac{1}{2} \mathbf{I} \end{bmatrix}
 \end{aligned} \tag{85}$$

其中 $[\cdot]_L$ 和 $[\cdot]_R$ 为四元数转为左/右旋转矩阵的算子。

(2) 角度误差对 j 时刻姿态求导

$$\begin{aligned}
\frac{\partial \mathbf{r}_q}{\partial \delta \boldsymbol{\theta}_{b_j b'_j}} &= \frac{\partial 2[\mathbf{q}_{b_i b_j}^* \otimes \mathbf{q}_{wb_i}^* \otimes \mathbf{q}_{wb_j} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_j b'_j} \end{bmatrix}]_{xyz}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
&= \frac{\partial 2[(\mathbf{q}_{b_i b_j}^* \otimes \mathbf{q}_{wb_i}^* \otimes \mathbf{q}_{wb_j})_L \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta}_{b_j b'_j} \end{bmatrix}]_{xyz}}{\partial \delta \boldsymbol{\theta}_{b_i b'_i}} \\
&= 2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} (\mathbf{q}_{b_i b_j}^* \otimes \mathbf{q}_{wb_i}^* \otimes \mathbf{q}_{wb_j})_L \begin{bmatrix} \mathbf{0} \\ \frac{1}{2} \mathbf{I} \end{bmatrix}
\end{aligned} \tag{86}$$

(3) 角度误差对 i 时刻陀螺仪偏置 \mathbf{b}_i^g

$$\begin{aligned}
\frac{\partial \mathbf{r}_q}{\partial \delta \mathbf{b}_i^g} &= \frac{\partial 2[(\mathbf{q}_{b_i b_j} \otimes \left[\frac{1}{2} \mathbf{J}_{b_i^g}^q \delta \mathbf{b}_i^g \right])^* \otimes \mathbf{q}_{wb_i}^* \otimes \mathbf{q}_{wb_j}]_{xyz}}{\partial \delta \mathbf{b}_i^g} \\
&= \frac{\partial - 2[((\mathbf{q}_{b_i b_j} \otimes \left[\frac{1}{2} \mathbf{J}_{b_i^g}^q \delta \mathbf{b}_i^g \right])^* \otimes \mathbf{q}_{wb_i}^* \otimes \mathbf{q}_{wb_j})^*]_{xyz}}{\partial \delta \mathbf{b}_i^g} \\
&= \frac{\partial - 2[\mathbf{q}_{wb_j}^* \otimes \mathbf{q}_{wb_i} \otimes (\mathbf{q}_{b_i b_j} \otimes \left[\frac{1}{2} \mathbf{J}_{b_i^g}^q \delta \mathbf{b}_i^g \right])]_{xyz}}{\partial \delta \mathbf{b}_i^g} \\
&= -2 \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} [\mathbf{q}_{wb_j}^* \otimes \mathbf{q}_{wb_i} \otimes \mathbf{q}_{b_i b_j}]_L \begin{bmatrix} \mathbf{0} \\ \frac{1}{2} \mathbf{J}_{b_i^g}^q \end{bmatrix}
\end{aligned} \tag{87}$$

作业

- 1 样例代码给出了使用 LM 算法来估计曲线 $y = \exp(ax^2 + bx + c)$ 参数 a, b, c 的完整过程。
 - ① 请绘制样例代码中 LM 阻尼因子 μ 随着迭代变化的曲线图
 - ② 将曲线函数改成 $y = ax^2 + bx + c$ ，请修改样例代码中残差计算，雅克比计算等函数，完成曲线参数估计。
 - ③ 实现其他更优秀的阻尼因子策略，并给出实验对比（选做，评优秀），策略可参考论文^a 4.1.1 节。
- 2 公式推导，根据课程知识，完成 F, G 中如下两项的推导过程：

$$\mathbf{f}_{15} = \frac{\partial \alpha_{b_i b_{k+1}}}{\partial \delta \mathbf{b}_k^g} = -\frac{1}{4} (\mathbf{R}_{b_i b_{k+1}} [(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)] \times \delta t^2) (-\delta t)$$

$$\mathbf{g}_{12} = \frac{\partial \alpha_{b_i b_{k+1}}}{\partial \mathbf{n}_k^g} = -\frac{1}{4} (\mathbf{R}_{b_i b_{k+1}} [(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)] \times \delta t^2) \left(\frac{1}{2} \delta t\right)$$

3 证明式(9)。

^aHenri Gavin. "The Levenberg-Marquardt method for nonlinear least squares curve-fitting problems". In: *Department of Civil and Environmental Engineering, Duke University* (2011), pp. 1–15.