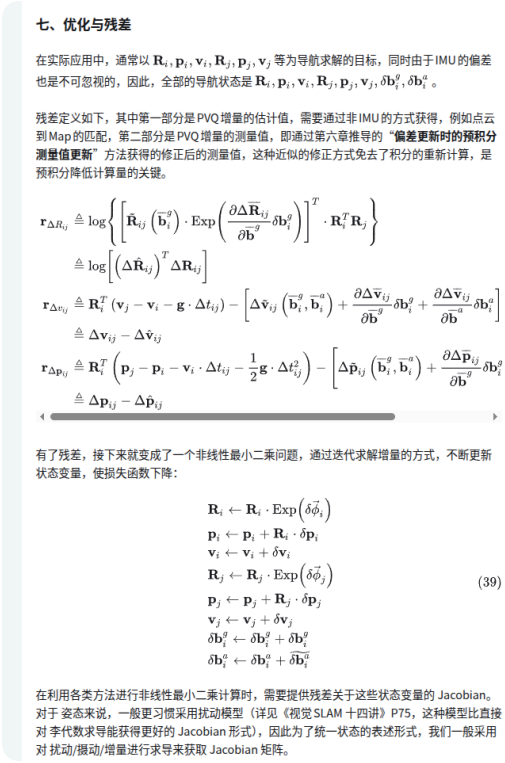

七、优化与残差

在实际应用中，通常以 $\mathbf{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{R}_j, \mathbf{p}_j, \mathbf{v}_j$ 等为导航求解的目标，同时由于IMU的偏差也是不可忽视的，因此，全部的导航状态是 $\mathbf{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{R}_j, \mathbf{p}_j, \mathbf{v}_j, \delta\mathbf{b}_i^g, \delta\mathbf{b}_i^a$ 。

残差定义如下，其中第一部分是PVQ增量的估计值，需要通过非IMU的方式获得，例如点云到Map的匹配，第二部分是PVQ增量的测量值，即通过第六章推导的 **“偏差更新时的预积分测量值更新”** 方法获得的修正后的测量值，这种近似的修正方式免去了积分的重新计算，是预积分降低计算量的关键。

$$
\begin{aligned}
\mathbf{r}_{\Delta R_{ij}} &\triangleq \log\left\{\left[\hat{\mathbf{R}}_{ij}\left(\overline{\mathbf{b}}_i^g\right)\cdot \mathrm{Exp}\left(\frac{\partial \Delta\overline{\mathbf{R}}_{ij}}{\partial \overline{\mathbf{b}}^g}\delta\mathbf{b}_i^g\right)\right]^T \cdot \mathbf{R}_i^T\mathbf{R}_j\right\} \\
&\triangleq \log\left[\left(\Delta\hat{\mathbf{R}}_{ij}\right)^T\Delta\mathbf{R}_{ij}\right] \\
\mathbf{r}_{\Delta v_{ij}} &\triangleq \mathbf{R}_i^T\left(\mathbf{v}_j-\mathbf{v}_i-\mathbf{g}\cdot\Delta t_{ij}\right)-\left[\Delta\hat{\mathbf{v}}_{ij}\left(\overline{\mathbf{b}}_i^g,\overline{\mathbf{b}}_i^a\right)+\frac{\partial\Delta\overline{\mathbf{v}}_{ij}}{\partial\overline{\mathbf{b}}^g}\delta\mathbf{b}_i^g+\frac{\partial\Delta\overline{\mathbf{v}}_{ij}}{\partial\overline{\mathbf{b}}^a}\delta\mathbf{b}_i^a\right] \\
&\triangleq \Delta\mathbf{v}_{ij}-\Delta\hat{\mathbf{v}}_{ij} \\
\mathbf{r}_{\Delta\mathbf{p}_{ij}} &\triangleq \mathbf{R}_i^T\left(\mathbf{p}_j-\mathbf{p}_i-\mathbf{v}_i\cdot\Delta t_{ij}-\frac{1}{2}\mathbf{g}\cdot\Delta t_{ij}^2\right)-\left[\Delta\hat{\mathbf{p}}_{ij}\left(\overline{\mathbf{b}}_i^g,\overline{\mathbf{b}}_i^a\right)+\frac{\partial\Delta\overline{\mathbf{p}}_{ij}}{\partial\overline{\mathbf{b}}^g}\delta\mathbf{b}_i^g\right. \\
&\triangleq \Delta\mathbf{p}_{ij}-\Delta\hat{\mathbf{p}}_{ij}
\end{aligned}
$$

有了残差，接下来就变成了一个非线性最小二乘问题，通过迭代求解增量的方式，不断更新状态变量，使损失函数下降：

$$
\begin{aligned}
\mathbf{R}_i &\leftarrow \mathbf{R}_i\cdot\mathrm{Exp}\left(\delta\vec{\phi}_i\right) \\
\mathbf{p}_i &\leftarrow \mathbf{p}_i+\mathbf{R}_i\cdot\delta\mathbf{p}_i \\
\mathbf{v}_i &\leftarrow \mathbf{v}_i+\delta\mathbf{v}_i \\
\mathbf{R}_j &\leftarrow \mathbf{R}_j\cdot\mathrm{Exp}\left(\delta\vec{\phi}_j\right) \\
\mathbf{p}_j &\leftarrow \mathbf{p}_j+\mathbf{R}_j\cdot\delta\mathbf{p}_j \\
\mathbf{v}_j &\leftarrow \mathbf{v}_j+\delta\mathbf{v}_j \\
\delta\mathbf{b}_i^g &\leftarrow \delta\mathbf{b}_i^g+\delta\mathbf{b}_i^g \\
\delta\mathbf{b}_i^a &\leftarrow \delta\mathbf{b}_i^a+\delta\overline{\mathbf{b}_i^a}
\end{aligned}
\tag{39}
$$

在利用各类方法进行非线性最小二乘计算时，需要提供残差关于这些状态变量的 Jacobian。对于 姿态来说，一般更习惯采用扰动模型（详见《视觉SLAM十四讲》P75，这种模型比直接对 李代数求导能获得更好的 Jacobian 形式），因此为了统一状态的表述形式，我们一般采用对 扰动/摄动/增量进行求导来获取 Jacobian 矩阵。

这部分我不太理解，如何理解优化的残差，有了残差后如何优化

下面我尽量用**直观+数学分步骤**的方式帮你彻底讲清楚这一页内容：
“残差怎么来的？为什么要这样写？有了残差后优化到底怎么做？”

🔎 1. 为什么需要 残差（Residual）

优化的目标就是：

> 让 IMU 预积分预测的相对状态（Ri, pj, vj 等）和实际估计的状态一致。

IMU 给的是**相对测量值**（ΔRij, Δpij, Δvij），但它是有偏差的（bias），并且噪声叠加后不准，所以我们要做的就是：

> **比较“预测的相对运动”和“IMU 预积分测量值”，得到差异，这个差异就是残差。**
> 优化 = 不断调整状态，使这个差异最小。

🔧 2. 残差是怎么推出来的？（逐个解释公式含义）

这部分残差由三类构成：

---

① 旋转残差 r_r

公式：

$$r_{\Delta R_{ij}} = \log\left(R_i^T R_j \cdot \Exp\left(\frac{\partial \Delta R_{ij}}{\partial b_g^i} \delta b_g^i \right)^T \cdot \Delta R_{ij}^T \right)$$

解释如下：

- **R_i^T R_j**：根据当前状态估计出的实际旋转变化
- **ΔR_ij**：IMU 预积分得到的 “测量的旋转变化”
- 残差 = 实际变化 × 测量变化 的差异
- log(·) 把 SO(3) 的旋转差变成向量（最小二乘必须用向量误差）

一句话理解：

r_R = “当前估计的旋转变化” - “预积分测量的旋转变化”

---

② 速度残差 r^v

$$r_{\Delta v_{ij}} = R_i^T (v_j - v_i - g\Delta t_{ij}) - (\Delta v_{ij} + \text{bias 修正项})$$

直观含义：

- 左边：根据当前状态算出来的速度变化
- 右边：IMU 预积分告诉你的速度变化（包含 bias 模型）

一句话理解：

r_v = “实际速度增量” - “IMU 预积分速度增量”

---

③ 位置残差 r^p

$$r_{\Delta p_{ij}} = R_i^T (p_j - p_i - v_i \Delta t_{ij} - \tfrac{1}{2} g \Delta t^2) - (\Delta p_{ij} + \text{bias 修正项})$$

解释：

- 左边：利用当前状态算出来的位移
- 右边：IMU 预积分测量的位移

一句话理解：

r_p = “实际位移增量” - “IMU 预积分位移增量”

📌 3. 为什么要加关于 bias 的偏导项?

预积分测量 ΔR_ij, Δv_ij, Δp_ij 与 bias 有关。

但我们 **不能每次优化都重新做一遍预积分**（太耗时间）。
所以论文中使用：

一阶泰勒展开逼近 “修正后的预积分量”
ΔR ≈ ΔR + (∂ΔR/∂bg) δbg

因此：

- 优化变量发生变化
- 只需要用偏导更新预积分，不需要重新积分

非常高效，这是 MSCKF/VIO/轮式惯导常用技巧。

🚀 4. 有了残差之后，如何优化?

下面是最重要的部分 —— **优化流程**。

步骤 1：定义优化变量（状态）

包括：

- 姿态 Ri
- 位置 pi
- 速度 vi
- IMU 偏差 b_g, b_a

这就是图优化中的节点。

步骤 2：构建残差向量

把前面三个残差拼成：

$$\mathbf{r}_{ij} = [r_{\Delta R},\ r_{\Delta v},\ r_{\Delta p}]$$

这是因子(graph factor)。

步骤 3：对每个变量计算 Jacobian

优化用的是 Gauss-Newton 或 Levenberg–Marquardt：

$$\delta x = -(J^T W J)^{-1} J^T W r$$

其中：

- **J** 是残差对状态变量的 Jacobian
- **r** 是残差
- **W** 是噪声协方差逆

论文末尾公式（39）就是更新规则，例如：

$$R_i \leftarrow R_i \cdot \mathrm{Exp}(\delta\phi_i)$$

$$p_i \leftarrow p_i + \delta p_i$$

这是典型的 “李群上增量更新”。

步骤 4：更新状态

用（39）更新所有状态变量。

步骤 5：重复迭代直到残差最小

整个过程就是：

> 预测 → 与预积分对比算残差 → 线性化 → 求增量 → 更新 → 继续循环
> 直到误差足够小

这就是 IMU 预积分在 VIO / SLAM 中的完整优化流程。

🎯 总结（最关键的几句话）

1. **残差 = 当前估计的相对运动 – IMU 预积分测量的相对运动**
2. 残差分三类：旋转、速度、位置
3. bias 修正项用于避免重新预积分
4. 优化流程：
   - 构建残差
   - 计算 Jacobian
   - Gauss-Newton 求增量
   - 用李群更新状态
   - 迭代

如果你愿意，我可以进一步提供：

✅ 把每一个残差公式画成直观示意图
✅ 用 C++/Python 给你写简化版 preintegration + 优化 demo
✅ 给你写 GTSAM 风格的伪代码，让你更容易理解
✅ 对照《VINS-Mono》源码讲这部分实际如何实现

你想继续往那一方向深入？

如何做优化，是对每个优化变量求导吗

是的，但不是“对每个变量单独求偏导那么简单”。
优化的核心是**构建残差 → 对所有优化变量整体求 Jacobian → 用 LM/GN 解一次线性系统 → 更新状态**。

我给你以 IMU 预积分为例，把“如何做优化”讲清楚：

🔥 1. 是不是对每个优化变量求导？

是的，但方式是：

每个残差项对每个参与该残差的状态变量求偏导（构成 Jacobian 矩阵）

比如某个 IMU 因子使用的状态是：

- i 时刻：Ri, pi, vi, bgi, bai
- j 时刻：Rj, pj, vj, bgj, baj（一般背景下 bg/bias 只用 i）

那么该因子的残差向量 r ∈ R⁹，会有 Jacobian：

$$J = \frac{\partial r}{\partial x}$$

其中 x = [Ri, pi, vi, bgi, bai, Rj, pj, vj]（通常 bias 只算一次）

🔧 2. Jacobian 是什么样子?

比如旋转残差：

$$r_R = \log((R_i^T R_j)\Delta R_{ij}^T)$$

它要对以下求导：

- ∂r_R / ∂Ri
- ∂r_R / ∂Rj
- ∂r_R / ∂bgi（因为 ΔRij 与 bias 有关）

速度残差：

$$r_v = R_i^T(v_j - v_i - g\Delta t) - (\Delta v_{ij} + J_{vbg}\delta bg + J_{vba}\delta ba)$$

要对：

- vi
- vj
- Ri
- bgi, bai
- Δvij（包含在预积分 Jacobian）

求导。

位置残差同理。

⚡ 3. 最终会得到一个很大的 Jacobian 矩阵

举例：

假设残差 r_ij 维度 9（3旋转+3速度+3位置）
状态变量维度 15（R,p,v,bg,ba）

则 J 的形状：

$$J_{ij} \in \mathbb{R}^{9\times 15}$$

每个因子都产生一个这样的 J。

在整个 SLAM 图中，会有很多因子：

$$J = \begin{bmatrix} J_{\text{imu}(0,1)} \\ J_{\text{imu}(1,2)} \\ J_{\text{imu}(2,3)} \\ \vdots \\ J_{\text{vision}} \\ J_{\text{prior}} \end{bmatrix}$$

这是总 Jacobian。

🚀 4. 用 Gauss–Newton 或 Levenberg–Marquardt 解优化

优化问题是：

$$\min_x \sum r(x)^T W r(x)$$

线性化后：

$$J^T W J \delta x = -J^T W r$$

这叫 **正规方程（Normal Equation）**。

通过求解 δx 得到对状态的增量。

一般使用：

- QR 分解

- Cholesky
- PCG（稀疏矩阵）

如 GTSAM 使用增量推理 + 因子图。

🔄 5. 用李群更新状态

旋转：

$$R \leftarrow R \cdot \mathrm{Exp}(\delta\phi)$$

向量变量：

$$p \leftarrow p + \delta p$$
$$v \leftarrow v + \delta v$$

bias：

$$bg \leftarrow bg + \delta bg$$
$$ba \leftarrow ba + \delta ba$$

然后进入下一轮迭代。

🧠 6. 总结一句话

> **优化不是一次对单个变量求导，而是对每个残差项分别对所有参与变量求偏导，组成整个系统的 Jacobian，利用 Gauss-Newton / LM 求一次线性系统，再用李群更新状态，迭代至收敛。**

如果你愿意，我可以继续给你：

📌 **具体 Jacobian 推导（含旋转的李群求导）**
📌 **给你一份"IMU 预积分因子的 Jacobian 表"**（类似 VINS/GTSAM）
📌 **写一个超简化 C++ 版本的优化 demo**
📌 **把整个优化过程画成流程图**

你想看哪一个？