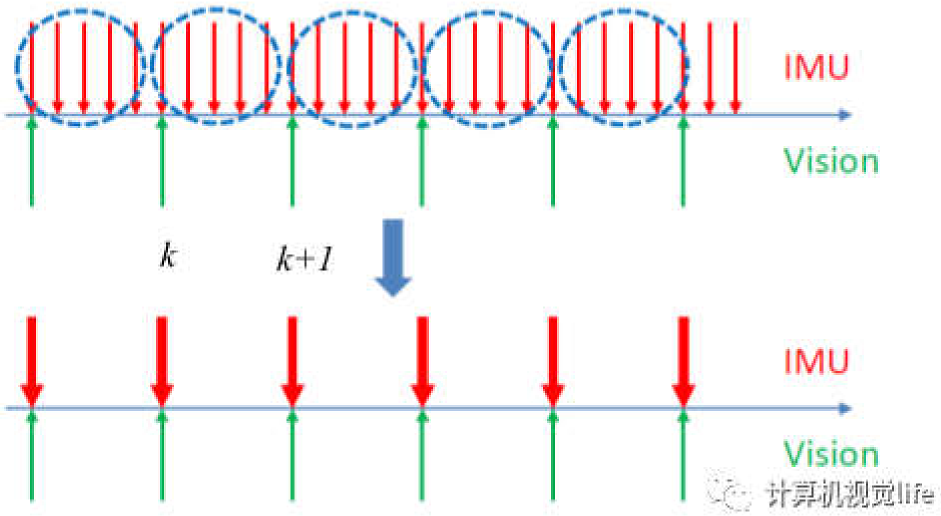
VINS 中的 IMU 预积分推导和代码解读

VIO 中,如果在世界坐标系中对 IMU 进行积分,积分项中包含体坐标系相对于世界坐标系的瞬时旋转矩阵。然而,在优化位姿时,关键帧时刻体坐标系相对于世界坐标系的旋转矩阵会发生变化,那么需要对 IMU 重新进行积分。预积分就是为了避免这种重复积分。IMU 预积分将参考坐标系改为前一帧的体坐标系,从而积出了两帧之间的相对运动。

预积分

将第 k 帧和第 k+1 帧之间的所有 IMU 进行积分,可得第 k+1 帧的位置、速度和旋转(PVQ),作为视觉估计的初始值，示意图如下:



从 IMU 获取body坐标系下的加速度计测量信息$\hat{\mathbf{a}}^b$和陀螺仪测量信息$\hat{\omega}^b$:

$$
\begin{aligned}
\hat{\omega}^b &= \omega^b + \mathbf{b}^g + \mathbf{n}^g \\
\hat{\mathbf{a}}^b &= \mathbf{a}^b + \mathbf{q}_w^b \mathbf{g}^w + \mathbf{b}^a + \mathbf{n}^a
\end{aligned} \tag{1}
$$

上式中，ω^b、a^b 为加速度计和陀螺仪的真值，$\mathbf{b}^g$、$\mathbf{n}^g$ 为陀螺仪的偏置和噪声，b^a、n^a 为加速的计的偏置的噪声，$\mathbf{q}_w^b$ 为世界坐标系到body坐标系的旋转四元数，$\mathbf{g}^w$ 为世界坐标系下的重力，**其符号的正负视坐标系而定**；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。

假设噪声$\mathbf{n}^g$、$\mathbf{n}^a$ 服从高斯分布：

$$
\begin{aligned}
\mathbf{n}^g &\sim N(0, \sigma_g^2) \\
\mathbf{n}^a &\sim N(0, \sigma_a^2)
\end{aligned} \tag{2}
$$

加速度偏置 $\mathbf{n}^a$和陀螺仪偏置$\mathbf{b}^g$被建模为随机游走，其导数为高斯性的，：

$$
\begin{aligned}
\dot{\mathbf{b}}^g &= \mathbf{n}_b{}^g \\
\dot{\mathbf{b}}^a &= \mathbf{n}_b{}^a
\end{aligned} \tag{3}
$$

PVQ对时间的导数可写成：

$$
\begin{aligned}
\dot{\mathbf{P}}_{b_t}^w &= \mathbf{v}_t^w \\
\dot{\mathbf{v}}_t^w &= \mathbf{a}_t^w \\
\dot{\mathbf{q}}_{b_t}^w &= \mathbf{q}_{b_t}^w \otimes \begin{bmatrix} 0 \\ \frac{1}{2}\omega^{b_t} \end{bmatrix}
\end{aligned} \tag{4}
$$

（点击图片放大看公式）

连续形式

对于连续两个关键帧 b_k 和 b_{k+1} ,它们对应的时刻分别为 t_k、t_{k+1} ，从第t_k 时刻的 PVQ 对 IMU 的测量值进行积分得到第 t_{k+1} 时刻的 PVQ：

$$\begin{aligned}
\mathbf{p}_{b_{k+1}}^w &= \mathbf{p}_{b_k}^w + \mathbf{v}_t^w \Delta t + \iint_{t\in[t_k,t_{k+1}]} (\mathbf{q}_{b_t}^w(\hat{\mathbf{a}}^{b_t} - \mathbf{b}_t^a) - \mathbf{g}^w)\delta t^2 \\
\mathbf{v}_{b_{k+1}}^w &= \mathbf{v}_{b_k}^w + \int_{t\in[t_k,t_{k+1}]} (\mathbf{q}_{b_t}^w(\hat{\mathbf{a}}^{b_t} - \mathbf{b}_t^a) - \mathbf{g}^w)\delta t \\
\mathbf{q}_{b_{k+1}}^w &= \int_{t\in[t_k,t_{k+1}]} \mathbf{q}_{b_t}^w \otimes \begin{bmatrix} 0 \\ \frac{1}{2}(\omega^{b_t} - \mathbf{b}_t^g) \end{bmatrix} \delta t
\end{aligned} \tag{5}$$

观察上式，再回到文章的最开始所述：此时积分项中包含体坐标系相对于世界坐标系的瞬时旋转矩阵，而在优化的过程中，关键帧相对于世界坐标系的位姿会发生变化，那么公式(5) 则需要重新积分，为避免重复积分，我们可以将关键帧到世界坐标系的变换通过公式(6) 进行转换，使积分项则变成相对于第 k 时刻的姿态，而不是相对于世界坐标系的姿态：

$$\mathbf{q}_{b_t}^w = q_{b_k}^w \otimes q_{b_t}^{b_k} \tag{6}$$

代入公式(5) 得：

$$\begin{aligned}
\mathbf{p}_{b_{k+1}}^w &= \mathbf{p}_{b_k}^w + \mathbf{v}_t^w \Delta t - \frac{1}{2}\mathbf{g}^w \Delta t^2 + \mathbf{q}_{b_k}^w \iint_{t\in[t_k,t_{k+1}]} (\mathbf{q}_{b_t}^{b_k}(\hat{\mathbf{a}}^{b_t} - \mathbf{b}_t^a)\delta t^2 \\
\mathbf{v}_{b_{k+1}}^w &= \mathbf{v}_{b_k}^w - \mathbf{g}^w \Delta t + \mathbf{q}_{b_k}^w \int_{t\in[t_k,t_{k+1}]} (\mathbf{q}_{b_t}^{b_k}(\hat{\mathbf{a}}^{b_t} - \mathbf{b}_t^a))\delta t \\
\mathbf{q}_{b_{k+1}}^w &= \mathbf{q}_{b_k}^w \int_{t\in[t_k,t_{k+1}]} \mathbf{q}_{b_t}^{b_k} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}(\omega^{b_t} - \mathbf{b}_t^g) \end{bmatrix} \delta t
\end{aligned} \tag{7}$$

公式(7) 中的积分项中的参考坐标系变成了 b_k ，积分结果为b_{k+1} 对于 b_k 的相对运动量，这样可以使在优化过程中即使对关键帧的位置、速度和旋转等状态进行调整，也不对积分项产生任何影响，从而避免了重复积分。

进一步整理：

$$\begin{aligned}
\alpha_{b_{k+1}}^{b_k} &= \iint_{t\in[t_k,t_{k+1}]} (\mathbf{q}_{b_t}^{b_k}(\hat{\mathbf{a}}^{b_t} - \mathbf{b}_t^a))\delta t^2 \\
\beta_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,t_{k+1}]} (\mathbf{q}_{b_t}^{b_k}(\hat{\mathbf{a}}^{b_t} - \mathbf{b}_t^a)))\delta t \\
\mathbf{q}_{b_{k+1}}^{b_k} &= \int_{t\in[t_k,t_{k+1}]} \mathbf{q}_{b_t}^{b_k} \otimes \begin{bmatrix} 0 \\ \frac{1}{2}(\omega^{b_t} - \mathbf{b}_t^g) \end{bmatrix} \delta t
\end{aligned} \tag{8}$$

从公式(8) 可知，预积分量不仅跟 IMU 测量值有关，还与IMU 的偏置是相关的，而偏置也是我们需要优化的变量，我们假设短时间内IMU的偏置是不变的，重新整理PVQ公式，有：

$$
\begin{bmatrix} \mathbf{p}_{k+1}^{w} \\ \mathbf{v}_{k+1}^{w} \\ \mathbf{q}_{k+1}^{w} \\ \mathbf{b}_{k+1}^{a} \\ \mathbf{b}_{k+1}^{g} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{b_k}^{w} + \mathbf{v}_{t}^{w}\Delta t - \frac{1}{2}\mathbf{g}^{w}\Delta t^2 + \mathbf{q}_{b_k}^{w}\alpha_{b_{k+1}}^{b_k} \\ \mathbf{v}_{b_k}^{w} - \mathbf{g}^{w}\Delta t + \mathbf{q}_{b_k}^{w}\beta_{b_{k+1}}^{b_k} \\ \mathbf{q}_{b_k}^{w} \otimes \mathbf{q}_{b_{k+1}}^{b_k} \\ \mathbf{b}_{k}^{a} \\ \mathbf{b}_{k}^{g} \end{bmatrix} \tag{9}
$$

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，得到预计分误差：

$$
\begin{bmatrix} \mathbf{r}_p \\ \mathbf{r}_v \\ \mathbf{r}_q \\ \mathbf{r}_{b^a} \\ \mathbf{r}_{b^g} \end{bmatrix}_{15\times 1} = \begin{bmatrix} \mathbf{q}_{w}^{b_k}(\mathbf{p}_{k+1}^{w} - \mathbf{p}_{b_k}^{w} - \mathbf{v}_{t}^{w}\Delta t + \frac{1}{2}\mathbf{g}^{w}\Delta t^2) - \alpha_{b_{k+1}}^{b_k} \\ \mathbf{q}_{w}^{b_k}(\mathbf{v}_{b_{k+1}}^{w} - \mathbf{v}_{b_k}^{w} + \mathbf{g}^{w}\Delta t) - \beta_{b_{k+1}}^{b_k} \\ 2(\mathbf{q}_{b_k}^{b_{k+1}} \otimes \mathbf{q}_{w}^{b_k} \otimes \mathbf{q}_{b_{k+1}}^{w})_{xyz} \\ \mathbf{b}_{k+1}^{a} - \mathbf{b}_{k}^{a} \\ \mathbf{b}_{k+1}^{g} - \mathbf{b}_{k}^{g} \end{bmatrix} \tag{10}
$$

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

离散形式

离散形式的有两种方法：欧拉法和中值法，作者论文中采用的欧拉法，代码实践中采用的是中值法，本文主要是对代码的解读，为与代码对应，所以只对中值法进行介绍。

PVQ模型

在开始时，$\alpha_{b_k}^{b_k}$、$\beta_{b_k}^{b_k}$是0，$\mathbf{q}_{b_k}^{b_k}$是单位四元数，α、β、$\mathbf{q}$在公式（8）是逐步传递的。另外，增加的噪声项$\mathbf{n}^a$, $\mathbf{n}^g$是未知的，在实现中被视为零，中值积分采用两个相邻时刻 k 到 k+1 的位姿是用两个时刻的测量值 a, ω 的平均值来计算，那么，可得PVQ公式：

$$
\begin{aligned}
\overline{\omega} &= \frac{1}{2}((\omega^{b_k} - \mathbf{b}_k^g) + (\omega^{b_{k+1}} - \mathbf{b}_k^g)) \\
\mathbf{q}_{b_{k+1}}^{b_i} &= \mathbf{q}_{b_k}^{b_i} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\overline{\omega}\delta t \end{bmatrix} \\
\overline{\mathbf{a}} &= \frac{1}{2}(\mathbf{q}_{b_k}^{b_i}(\mathbf{a}^{b_k} - \mathbf{b}_k^a) + \mathbf{q}_{b_{k+1}}^{b_i}(\mathbf{a}^{b_{k+1}} - \mathbf{b}_k^a)) \\
\alpha_{b_{k+1}}^{b_i} &= \alpha_{b_k}^{b_i} + \beta_{b_k}^{b_i}\delta t + \frac{1}{2}\overline{\mathbf{a}}\delta t^2 \\
\beta_{b_{k+1}}^{b_i} &= \beta_{b_k}^{b_i} + \overline{\mathbf{a}}\delta t \\
\mathbf{b}_{k+1}^{a} &= \mathbf{b}_{k}^{a} + \mathbf{n}_{b_k^a}\delta t \\
\mathbf{b}_{k+1}^{g} &= \mathbf{b}_{k}^{g} + \mathbf{n}_{b_k^g}\delta t
\end{aligned} \tag{11}
$$

其中，i是在$[t_k, t_{k+1}]$ 中IMU测量值对应的离散时刻，δt是IMU测量值k和$k+1$ 之间的时间间隔。

对应代码如下：

```
//采用的是中值积分的传播方式
Vector3d un_gyr = 0.5 * (gyr_0 + angular_velocity) - Bgs[j];
Rs[j] *= Utility::deltaQ(un_gyr * dt).toRotationMatrix();
Vector3d un_acc_1 = Rs[j] * (linear_acceleration - Bas[j]) - g;
Vector3d un_acc = 0.5 * (un_acc_0 + un_acc_1);
Ps[j] += dt * Vs[j] + 0.5 * dt * dt * un_acc;
Vs[j] += dt * un_acc;
```

（左右滑动试试）

误差传播

下面，考虑离散形式下的误差传递过程，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$\mathbf{n}$，对非线性系统状态方程$\mathbf{x}_{k+1}=f(\mathbf{x}_k,\mathbf{u}_k)$进行一阶泰勒展开有:

$$\begin{aligned}\mathbf{x}_{k+1}&=f(\mathbf{x}_k,\mathbf{u}_k)\\ \hat{\mathbf{x}}_{k+1}+\delta\mathbf{x}_{k+1}&=f(\hat{\mathbf{x}}_k+\delta\mathbf{x}_k,\hat{\mathbf{u}}_k+\mathbf{n}_k)\\ \hat{\mathbf{x}}_{k+1}+\delta\mathbf{x}_{k+1}&=f(\hat{\mathbf{x}}_k,\hat{\mathbf{u}}_k)+\mathbf{F}\delta\mathbf{x}_k+\mathbf{V}\mathbf{n}_k\end{aligned}\tag{12}$$

得：

$$\delta\mathbf{x}_{k+1}=\mathbf{F}\delta\mathbf{x}_k+\mathbf{V}\mathbf{n}_k\tag{13}$$

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

根据公式(11)，将测量噪声考虑进来，更改PVQ模型为：

$$\begin{aligned}\overline{\omega}&=\frac{1}{2}((\omega^{b_k}+\mathbf{n}_k^g-\mathbf{b}_k^g)+(\omega^{b_{k+1}}+\mathbf{n}_{k+1}^g-\mathbf{b}_k^g))\\ \mathbf{q}_{b_{k+1}}^{b_i}&=\mathbf{q}_{b_k}^{b_i}\otimes\begin{bmatrix}1\\ \frac{1}{2}\overline{\omega}\delta t\end{bmatrix}\\ \overline{\mathbf{a}}&=\frac{1}{2}(\mathbf{q}_{b_k}^{b_i}(\mathbf{a}^{b_k}+\mathbf{n}_{k+1}^a-\mathbf{b}_k^a)+\mathbf{q}_{b_{k+1}}^{b_i}(\mathbf{a}^{b_{k+1}}+\mathbf{n}_{k+1}^a-\mathbf{b}_k^a))\\ \alpha_{b_{k+1}}^{b_i}&=\alpha_{b_k}^{b_i}+\beta_{b_k}^{b_i}\delta t+\frac{1}{2}\overline{\mathbf{a}}\delta t^2\\ \beta_{b_{k+1}}^{b_i}&=\beta_{b_k}^{b_i}+\overline{\mathbf{a}}\delta t\\ \mathbf{b}_{k+1}^a&=\mathbf{b}_k^a+\mathbf{n}_{b_k^a}\delta t\\ \mathbf{b}_{k+1}^g&=\mathbf{b}_k^g+\mathbf{n}_{b_k^g}\delta t\end{aligned}\tag{14}$$

VIO系统中t_{k+1}时刻的状态量$\mathbf{q}_{b_{k+1}}^{b_i},\alpha_{b_{k+1}}^{b_i},\beta_{b_{k+1}}^{b_i},\mathbf{b}_{k+1}^a,\mathbf{b}_{k+1}^g$误差来源主要是$t_k$时刻的状态量$\mathbf{q}_{b_k}^{b_i},\alpha_{b_k}^{b_i},\beta_{b_k}^{b_i},\mathbf{b}_k^a,\mathbf{b}_k^g$的误差的传播，$t_k$、$t_{k+1}$时刻加速度测量值和角速度测量值的高斯白噪声，以及加速度和角速度随机游走bias的高斯白噪声。

由公式(13)、(14)得t_{k+1}误差有：

$$
\begin{bmatrix} \delta\alpha_{b_{k+1}} \\ \delta\theta_{b_{k+1}} \\ \delta\beta_{b_{k+1}} \\ \delta\mathbf{b}^a_{k+1} \\ \delta\mathbf{b}^g_{k+1} \end{bmatrix} = \mathbf{F} \begin{bmatrix} \delta\alpha_{b_k} \\ \delta\theta_{b_k} \\ \delta\beta_{b_k} \\ \delta\mathbf{b}^a_k \\ \delta\mathbf{b}^g_k \end{bmatrix}_{15\times1} + \mathbf{V} \begin{bmatrix} \mathbf{n}^a_k \\ \mathbf{n}^g_k \\ \mathbf{n}^a_{k+1} \\ \mathbf{n}^g_{k+1} \\ \mathbf{n}_{b^a_k} \\ \mathbf{n}_{b^g_k} \end{bmatrix}_{18\times1} \tag{15}
$$

F，**V** 为两个时刻间的协方差传递矩阵，我们直接给出具体形式，省略推导过程为：

$$
\mathbf{F} = \begin{bmatrix} \mathbf{I} & \mathbf{f}_{12} & \mathbf{I}\delta t & -\frac{1}{4}(\mathbf{q}^{b_i}_{b_k} + \mathbf{q}^{b_i}_{b_{k+1}})\delta t^2 & \mathbf{f}_{15} \\ \mathbf{0} & \mathbf{I} - [\omega]_\times & \mathbf{0} & \mathbf{0} & -\mathbf{I}\delta t \\ \mathbf{0} & \mathbf{f}_{32} & \mathbf{I} & -\frac{1}{2}(\mathbf{q}^{b_i}_{b_k} + \mathbf{q}^{b_i}_{b_{k+1}})\delta t & \mathbf{f}_{35} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}_{15\times15} \tag{16}
$$

$$
\mathbf{V} = \begin{bmatrix} \frac{1}{4}\mathbf{q}^{b_i}_{b_k}\delta t^2 & \mathbf{v}_{12} & \frac{1}{4}\mathbf{q}^{b_i}_{b_{k+1}}\delta t^2 & \mathbf{v}_{14} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\mathbf{I}\delta t & \mathbf{0} & \frac{1}{2}\mathbf{I}\delta t & \mathbf{0} & \mathbf{0} \\ \frac{1}{2}\mathbf{q}^{b_i}_{b_k}\delta t & \mathbf{v}_{32} & \frac{1}{2}\mathbf{q}^{b_i}_{b_{k+1}}\delta t & \mathbf{v}_{34} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\delta t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\delta t \end{bmatrix}_{15\times18} \tag{17}
$$

其中：

$$
\mathbf{f}_{12} = \frac{\partial\alpha^{b_i}_{b_{k+1}}}{\partial\delta\theta_{b_k}} = -\frac{1}{4}(\mathbf{q}^{b_i}_{b_k}[a^{b_k} - b^a_k]_\times\delta t^2 + \mathbf{q}^{b_i}_{b_{k+1}}[a^{b_{k+1}} - b^a_k]_\times(\mathbf{I} - [\omega]_\times\delta t)\delta t^2)
$$

$$
\mathbf{f}_{32} = \frac{\partial\beta^{b_i}_{b_{k+1}}}{\partial\delta\theta_{b_k}} = -\frac{1}{2}(\mathbf{q}^{b_i}_{b_k}[a^{b_k} - b^a_k]_\times\delta t + \mathbf{q}^{b_i}_{b_{k+1}}[a^{b_{k+1}} - b^a_k]_\times(\mathbf{I} - [\omega]_\times\delta t)\delta t)
$$

$$
\mathbf{f}_{15} = \frac{\partial\alpha^{b_i}_{b_{k+1}}}{\partial\delta\mathbf{b}^g_k} = -\frac{1}{4}(\mathbf{q}^{b_i}_{b_{k+1}}[a^{b_{k+1}} - b^a_k]_\times\delta t^2)(-\delta t)
$$

$$
\mathbf{f}_{35} = \frac{\partial\beta^{b_i}_{b_{k+1}}}{\partial\delta\mathbf{b}^g_k} = -\frac{1}{2}(\mathbf{q}^{b_i}_{b_{k+1}}[a^{b_{k+1}} - b^a_k]_\times\delta t)(-\delta t)
$$

$$
\mathbf{v}_{12} = \frac{\partial\alpha^{b_i}_{b_{k+1}}}{\partial\mathbf{n}^g_k} = \mathbf{v}_{14} = \frac{\partial\alpha^{b_i}_{b_{k+1}}}{\partial\mathbf{n}^g_{k+1}} = -\frac{1}{4}(\mathbf{q}^{b_i}_{b_{k+1}}[a^{b_{k+1}} - b^a_k]_\times\delta t^2)(\frac{1}{2}\delta t)
$$

$$
\mathbf{v}_{32} = \frac{\partial\beta^{b_i}_{b_{k+1}}}{\partial\mathbf{n}^g_k} = \mathbf{v}_{34} = \frac{\partial\beta^{b_i}_{b_{k+1}}}{\partial\mathbf{n}^g_{k+1}} = -\frac{1}{2}(\mathbf{q}^{b_i}_{b_{k+1}}[a^{b_{k+1}} - b^a_k]_\times\delta t^2)(\frac{1}{2}\delta t)
$$

下对应代码在integration_base.h文件的**midPointIntegration**()，

F:

```
MatrixXd F = MatrixXd::Zero(15, 15);
F.block<3, 3>(0, 0) = Matrix3d::Identity();
F.block<3, 3>(0, 3) = -0.25 * delta_q.toRotationMatrix() * R_a_0_x * _dt * _dt +
                      -0.25 * result_delta_q.toRotationMatrix() * R_a_1_x *
                      (Matrix3d::Identity() - R_w_x * _dt) * _dt * _dt;
F.block<3, 3>(0, 6) = MatrixXd::Identity(3,3) * _dt;
```

```
F.block<3, 3>(0, 9) = -0.25 * (delta_q.toRotationMatrix() +
                       result_delta_q.toRotationMatrix()) * _dt * _dt;
F.block<3, 3>(0, 12) = -0.25 * result_delta_q.toRotationMatrix() * R_a_1_x * _dt * _dt *
                       _dt;
F.block<3, 3>(3, 3) = Matrix3d::Identity() - R_w_x * _dt;
F.block<3, 3>(3, 12) = -1.0 * MatrixXd::Identity(3,3) * _dt;
F.block<3, 3>(6, 3) = -0.5 * delta_q.toRotationMatrix() * R_a_0_x * _dt +
                      -0.5 * result_delta_q.toRotationMatrix() * R_a_1_x *
                      (Matrix3d::Identity() - R_w_x * _dt) * _dt;
F.block<3, 3>(6, 6) = Matrix3d::Identity();
F.block<3, 3>(6, 9) = -0.5 * (delta_q.toRotationMatrix() +
                      result_delta_q.toRotationMatrix()) * _dt;
F.block<3, 3>(6, 12) = -0.5 * result_delta_q.toRotationMatrix() * R_a_1_x * _dt * -_dt;
F.block<3, 3>(9, 9) = Matrix3d::Identity();
F.block<3, 3>(12, 12) = Matrix3d::Identity();
```

（左右滑动试试）

V:

```
MatrixXd V = MatrixXd::Zero(15,18);
V.block<3, 3>(0, 0) =  0.25 * delta_q.toRotationMatrix() * _dt * _dt;
V.block<3, 3>(0, 3) =  0.25 * -result_delta_q.toRotationMatrix() * R_a_1_x  * _dt * _dt
                       0.5 * _dt;
V.block<3, 3>(0, 6) =  0.25 * result_delta_q.toRotationMatrix() * _dt * _dt;
V.block<3, 3>(0, 9) =  V.block<3, 3>(0, 3);
V.block<3, 3>(3, 3) =  0.5 * MatrixXd::Identity(3,3) * _dt;
V.block<3, 3>(3, 9) =  0.5 * MatrixXd::Identity(3,3) * _dt;
V.block<3, 3>(6, 0) =  0.5 * delta_q.toRotationMatrix() * _dt;
V.block<3, 3>(6, 3) =  0.5 * -result_delta_q.toRotationMatrix() * R_a_1_x  * _dt * 0.5 *
                       _dt;
V.block<3, 3>(6, 6) =  0.5 * result_delta_q.toRotationMatrix() * _dt;
V.block<3, 3>(6, 9) =  V.block<3, 3>(6, 3);
V.block<3, 3>(9, 12) = MatrixXd::Identity(3,3) * _dt;
V.block<3, 3>(12, 15) = MatrixXd::Identity(3,3) * _dt;
```

（左右滑动试试）

离散形式的 PVQ 增量误差的 Jacobian 和协方差

将公式(15) 简写为：

$$\delta \mathbf{z}_{k+1} = \mathbf{F} \delta \mathbf{z}_k + \mathbf{V}\mathbf{Q} \tag{18}$$

那么Jacobian和协防差的迭代公式为：

$$\begin{aligned} J_{k+1} &= \mathbf{F} J_k \\ P_{k+1} &= \mathbf{F} P_k \mathbf{F}^T + \mathbf{V}\mathbf{Q}\mathbf{V}^T \end{aligned} \tag{19}$$

其中$\mathbf{Q}$ 表示噪声项的对角协方差矩阵，由于假设各个分量相互独立，所以$\mathbf{Q}$ 为对角阵：

$$\mathbf{Q} = \begin{bmatrix} \sigma_{a_k}^2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_{g_k}^2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \sigma_{a_{k+1}}^2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_{g_{k+1}}^2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_{b_k^a}^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \sigma_{b_k^g}^2 \end{bmatrix} \tag{20}$$

对应代码在integration_base.h文件的**midPointIntegration**()：

```
jacobian = F * jacobian;
covariance = F * covariance * F.transpose() + V * noise * V.transpose();
```

有了公式(19)，同时知道了Jacobian初值$J_k = \mathbf{I}$和协方差的初值$P_k = 0$，我们就可以通过迭代求出后续时刻的Jacobian和协方差，而之所以要迭代求解Jacobian，是为了给后面提供给对 bias 的近似计算。

现在再回过头来看公式(14)，可以知道预积分的值是与bias相关的，而 bias 也是我们需要优化的变量，这将导致的问题是，当每次迭代时，我们得到一个新的 bias，又得根据公式(14)重新对第k帧和第$k+1$帧之间的 IMU 预积分，为避免重复积分增加计算消耗，我们假设预积分的变化量与 bias 是线性关系，所以对于预积分量直接在$k+1$时刻的 bias 附近用一阶泰勒展开来近似：

$$\begin{aligned} \alpha_{b_{k+1}}^{b_k} &= \hat{\alpha}_{b_{k+1}}^{b_k} + J_{b_k^a}^{\alpha} \delta \mathbf{b}_k^a + J_{b_k^g}^{\alpha} \delta \mathbf{b}_k^g \\ \beta_{b_{k+1}}^{b_k} &= \hat{\beta}_{b_{k+1}}^{b_k} + J_{b_k^a}^{\beta} \delta \mathbf{b}_k^a + J_{b_k^g}^{\beta} \delta \mathbf{b}_k^g \\ \mathbf{q}_{b_{k+1}}^{b_k} &= \hat{\mathbf{q}}_{b_{k+1}}^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} J_{b_k^g}^{q} \delta \mathbf{b}_k^g \end{bmatrix} \end{aligned} \tag{21}$$

其中$J_{b_k^a}^{\alpha} = \frac{\partial \alpha_{b_{k+1}}^{b_k}}{\partial \delta \mathbf{b}_k^a}, J_{b_k^g}^{\alpha} = \frac{\partial \alpha_{b_{k+1}}^{b_k}}{\partial \delta \mathbf{b}_k^g}, J_{b_k^a}^{\beta} = \frac{\partial \beta_{b_{k+1}}^{b_k}}{\partial \beta \mathbf{b}_k^a}, J_{b_k^g}^{\beta} = \frac{\partial \beta_{b_{k+1}}^{b_k}}{\partial \delta \mathbf{b}_k^g}, J_{b_k^g}^{q} = \frac{\mathbf{q}_{b_{k+1}}^{b_k}}{\partial \mathbf{b}_k^g}$ 表示预积分量对k时刻的 bias 求导。当对加速度计偏置和陀螺仪的偏置发生(微小)改变时,就可以根据公式(21)对预积分项进行修正,避免了重复积分。