VINS-MONO 论文及代码解析

李云天

2019 年 9 月 12 日

Harbin Institute of Technology

School of Astronautics

Technical Report No. HIT-2019-08

HARBIN INSTITUTE OF TECHNOLOGY

VINS-MONO 论文及代码解析

李云天

Harbin Institute of Technology
School of Astronautics
92 Xidazhi Street, Harbin, 150000
www.hit.edu.cn

2019 年 9 月 12 日

Technical Report No. ECE-2019-08

目录

插图

表格

1 概述

1.1 系统概览

单目相机和低成本 IMU 构成了能对空间六自由度状态进行估计的最小传感器单元——惯性视觉导航系统（Visual Inertial Navigation System，VINS）。但距离和位姿直接量测信息的缺乏，给 VINS 带来了 IMU 处理、估计器初始化、相机标定和非线性优化等方面的挑战。本文提出了一种鲁棒多功能单目视觉惯性导航系统——VINS-Mono。本方法基于鲁棒初始化和故障恢复，融合紧耦合非线性优化方法、IMU 预积分方法以及特征检测方法以获取高精度的视觉惯性里程计。回环检测模块的加入则进一步实现了紧耦合下最小计算量的重定位。此外，本方法还是用四方向位姿图优化加强全局一致性。算法的性能在公开数据集和实际飞行试验中得到了有效验证，并和现有算法进行了对比。单目相机的小尺寸、低成本、易配置等特点使其成为了 SLAM 领域的最常用传感器，但纯单目视觉系统无法恢复几何尺度；而 IMU 可以提供几何尺度、角速度等全面的量测信息，同时缓解由于光照变化、特征缺失、运动模糊等带来的视觉跟踪丢失问题。由于单目 VINS 需要加速度激励才能使几何尺度可观，单目 VINS 必须在未知的运动而非静止状态下完成初始化，这就给 VINS 中状态估计器的初始化带来了挑战。

VINS-Mono 正是为解决这一问题而诞生，本方法起始于状态估计器的**在线初始化**，核心是基于**非线性滑窗优化**的**紧耦合** VIO。该 VIO 模块不仅提供精确的局部位姿、速度和航向估计，还提供相机-IMU 内参和零偏的**在线修正**。回环检测则基于 **Dow2** 词袋实现。重定位基于单目 VIO 的**特征级融合**实现以实现最小计算复核下的精确鲁棒回环检测。同时 VINS-Mono 还加入了**几何回环**验证模块基于 IMU 的角度量测优化 **4 自由度位姿图**以保证全局一致性。

图 1-1给出了 VINS 的总体结构，系统起始于量测处理模块，包括图像特征的提取和跟踪，以及两图像之间 IMU 的预积分。随后初始化模块提供位姿、速度、重力矢量、陀螺偏差、3D 特征位置等必要的信息供后续基于非线性优化的 VIO 使用。带有重定位功能的紧耦合 VIO 模块负责融合预积分的 IMU 量测、图像特征量测以及回环中重新检测到的图像特征信息。最终，位子图优化基于几何验证重定位结果执行全局优化以消除飘逸。VIO、重定位和位姿图优化分别运行在独立的线程中以保证系统运行的可靠性和实时性。

1.2 符号约定

这里首先给出本文中的符号约定。a 等小写常规字母表示标量，$\boldsymbol{x}$ 等小写加粗字母表示矢量，$\boldsymbol{J}$ 等大写加粗字母表示矩阵。字母上标表示该标量所在的坐标系，$(\cdot)^w$ 为世界坐标系，采用 N-E-D 定义，z 轴指向重力矢量方向；$(\cdot)^b$ 为本体坐标系，采用前右下定义，IMU 测量坐标系于本体系一致；$(\cdot)^w$ 为相机坐标系，同样采用前右下定义，不同之处在于此处以相机光轴为前。坐标旋转分别采用旋转矩阵 $\boldsymbol{R}$ 和四元数 $\boldsymbol{q}$ 表示，旋转方向为下标到上标，即 $\boldsymbol{q}_b^w$ 表示本体系到世界系的坐标旋转四元数。$\boldsymbol{t}_b^w$ 表示本体系到世界系的坐标平移变换矢量。下标 k 表示当前对应的图像帧，例

如 c_k、b_k 分别表示第 k 帧图像采集时的本体坐标系和相机坐标系。$\boldsymbol{g} = [0, 0, g]^T$ 为重力矢量。$\otimes$ 为四元数乘法运算符。使用上标 $\hat{(\cdot)}$ 表示变量的估计值或测量值，以和实际值区分。

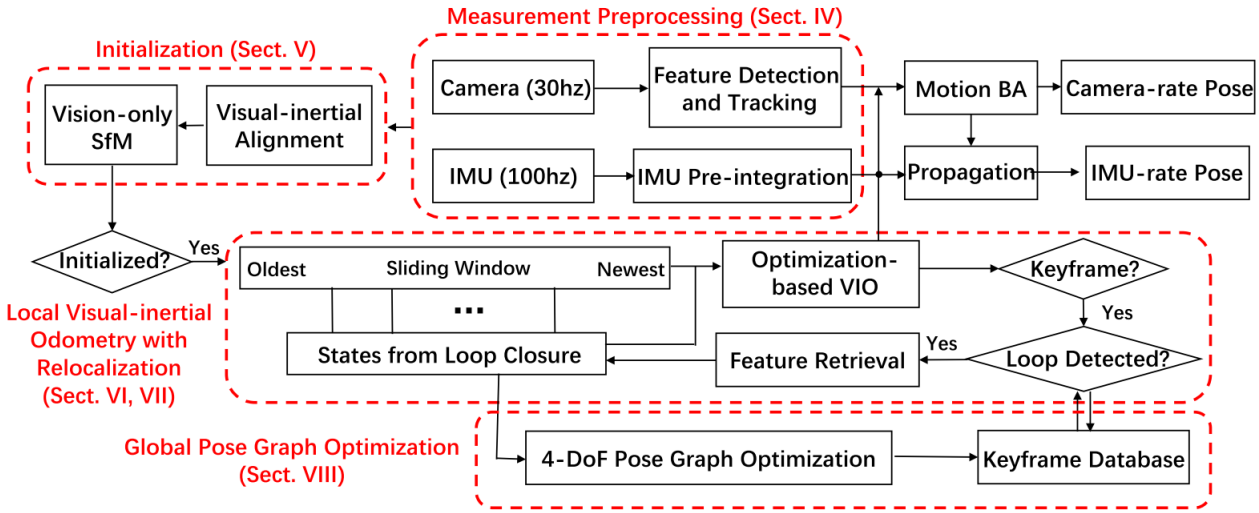

图 1-1: VINS-Mono 系统框架

2 量测处理

本节主要介绍连续两帧间 IMU 和相机的量测处理过程。相机量测处理分为跟踪参考帧关键点（Keypoints）在当前帧中的位置和探测当前帧中新的关键点两部分；IMU 量测的处理则主要是在集中在预积分过程。由于低成本 IMU 中零偏和噪声量级较为接近，这里假设噪声为高斯白噪声，在预积分过程中只考虑器件零偏的影响。

2.1 视觉前端处理

对于每一帧新图像，基于 KLT 稀疏光流法跟踪现有特征；同时采用 Shi-Tomasi 角点检测检测当前帧中新出现的特征以保证图像中的最小特征数（100-300）。同时，通过设置相邻特征之间的最小像素距离保证特征在图像中均匀分布。2D 特征首先经过去畸变处理，随后基于 RANSAC 算法去除野值量测，最后投影到单位球上。

关键帧的选取则基于以下两个原则。首先，判断两帧之间的平均视差，如果所跟踪的特征在当前帧和上一个参考帧之间的视差超过一定阈值，则将当前帧作为新的关键帧；同时为了分离纯旋转引起的视差变化，对两帧之间的陀螺量测进行积分，补偿旋转的影响。这一旋转补偿只作用于关键帧选取而给位姿估计，因此不会影响位姿估计质量。其次，判断所跟踪的特征数目，若小于一定数量，则将当前帧作为新的关键帧，以防跟踪丢失。

2.2 IMU 预积分

IMU 预积分基于连续时间四元数微分方法实现，相关公式的详细推导参见附录 B，这里直接给出结论。

2.2.1 IMU 器件模型

IMU 的量测模型可表示为:

$$\begin{cases} \hat{\boldsymbol{a}}^b = \boldsymbol{a}^b + \boldsymbol{b}_a + \boldsymbol{R}_w^b \boldsymbol{g}^w + \boldsymbol{n}_a \\ \hat{\boldsymbol{\omega}}^b = \boldsymbol{\omega}^b + \boldsymbol{b}_\omega + \boldsymbol{n}_\omega \end{cases} \tag{2-1}$$

式中，$\hat{\boldsymbol{a}}^b$ 和 $\hat{\boldsymbol{\omega}}^b$ 分别表示 IMU 的加速度和角速度测量信息；$\boldsymbol{a}^b$ 和 $\boldsymbol{\omega}^b$ 分别为载体真实的加速度和角速度；$\boldsymbol{b}_a$ 和 $\boldsymbol{b}_\omega$ 分别为加速度计和陀螺仪零偏，建模为随机游走过程:

$$\begin{cases} \boldsymbol{n}_{\boldsymbol{b}_a} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\sigma}_{b_a}^2\right), \boldsymbol{n}_{\boldsymbol{b}_\omega} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\sigma}_{b_\omega}^2\right) \\ \dot{\boldsymbol{b}_a} = \boldsymbol{n}_{\boldsymbol{b}_a}, dot\boldsymbol{b}_\omega = \boldsymbol{n}_{\boldsymbol{b}_\omega} \end{cases} \tag{2-2}$$

$\boldsymbol{R}_w^b$ 为世界坐标系到本体坐标系的坐标旋转矩阵；$\boldsymbol{g}^w$ 为世界坐标系下的重力加速度矢量；$\boldsymbol{n}_a \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\sigma}_a^2)$ 和 $\boldsymbol{n}_\omega \sim (\boldsymbol{0}, \boldsymbol{\sigma}_\omega^2)$ 为器件的高斯白噪声。

2.2.2 IMU 预积分项

由于 IMU 直接预积分项包含 k 时刻的位姿信息，不便于后端优化，因此使用增量形式的预积分项表达式:

$$\begin{cases} \hat{\boldsymbol{\alpha}}_{k+1}^{b_k} = \iint_{t_i \in [t_k, t_{k+1}]} \left(\boldsymbol{R}_{b_{t_i}}^w \left(\hat{\boldsymbol{a}}_{t_i} - \boldsymbol{b}_a - \boldsymbol{n}_a \right) \right) dt^2 \\ \hat{\boldsymbol{\beta}}_{k+1}^{b_k} = \int_{t_i \in [t_k, t_{k+1}]} \left(\boldsymbol{R}_{b_{t_i}}^w \left(\hat{\boldsymbol{a}}_{t_i} - \boldsymbol{b}_a - \boldsymbol{n}_a \right) \right) dt \\ \hat{\boldsymbol{\gamma}}_{k+1}^{b_k} = \int_{t_i \in [t_k, t_{k+1}]} \frac{1}{2} \boldsymbol{\Omega} \left(\hat{\boldsymbol{\omega}}_{wb}^w - \boldsymbol{b}_\omega - \boldsymbol{n}_\omega \right) \boldsymbol{\gamma}_k^{t_i} dt \end{cases} \tag{2-3}$$

式中 $\boldsymbol{\alpha}$、$\boldsymbol{\beta}$ 和 $\boldsymbol{\gamma}$ 分别为位置、速度和四元数的增量预积分项。

2.2.3 增量预积分关于零偏的更新

虽然增量形式的预积分避免了每次后端优化引起 k 时刻位姿变化时都需要重新预积分的问题，但仍需要根据零偏优化数的变化进行更新。为减少计算量，VINS-Mono 采取入夏更新策略: 当后端优化的 IMU 零偏变化时，若变化较小，则使用 IMU 预积分项关于零偏的一阶近似更新预

积分项；否则，则按照式 B-31重新积分 IMU 得到预积分项。IMU 预积分关于零偏的线性化更新表达式可以表示为：

$$\begin{cases} \hat{\boldsymbol{\alpha}}_k^{k+1} \approx \hat{\boldsymbol{\alpha}}_k^{k+1} + \boldsymbol{J}_{\boldsymbol{b}_a}^{\boldsymbol{\alpha}} \delta \boldsymbol{b}_a + \boldsymbol{J}_{\boldsymbol{b}_\omega}^{\boldsymbol{\alpha}} \delta \boldsymbol{b}_\omega \\ \hat{\boldsymbol{\beta}}_k^{k+1} \approx \hat{\boldsymbol{\beta}}_k^{k+1} + \boldsymbol{J}_{\boldsymbol{b}_a}^{\boldsymbol{\beta}} \delta \boldsymbol{b}_a + \boldsymbol{J}_{\boldsymbol{b}_\omega}^{\boldsymbol{\beta}} \delta \boldsymbol{b}_\omega \\ \hat{\boldsymbol{\gamma}}_k^{k+1} = \hat{\boldsymbol{\gamma}}_k^{k+1} \otimes \boldsymbol{J}_{\boldsymbol{b}_\omega}^{\boldsymbol{\gamma}} \delta \boldsymbol{b}_\omega \end{cases} \tag{2-4}$$

3 估计器初始化

单目紧耦合 VIO 是一个高度依赖于精确初始值的非线性系统，VINS-Mono 通过 IMU 预积分和视觉 SfM（Structure from Motion）获得的松耦合对准获得。

3.1 相机与 IMU 外参对准

与精确标定的数据集不同，在一些实际应用中，相机和 IMU 之间外参矩阵的标定精度不够。对于这种情况，VINS-Mono 将标定后的外参作为初值，在初始化过程中对其进行进一步优化。由于平移的影响并没有旋转显著，VINS-Mono 只考虑 IMU 与相机之间相对旋转矩阵的标定。

假设相机得到的两帧之间旋转矩阵为 $\boldsymbol{R}_{c_k}^{c_{k+1}}$，IMU 预积分得到的旋转矩阵为 $\boldsymbol{R}_{b_k}^{b_{k+1}}$，相机和 IMU 之间的旋转矩阵为 $\boldsymbol{R}_b^c$，则对于任意两帧图像，均有：

$$\boldsymbol{R}_{b_{k+1}}^{b_k} \boldsymbol{R}_{\boldsymbol{c}}^{\boldsymbol{b}} = \boldsymbol{R}_c^b \boldsymbol{R}_{c_{k+1}}^{c_k} \tag{3-5}$$

3.2 滑窗内的纯视觉 SfM

3.2.1 算法概览

为了限制算法的计算规模，前端在滑动窗口内仅维持有限数目的图像帧。首先，检测最后一帧图像和窗口内所有其他图像中的特征跟踪情况：若最后一帧图像和任意一个图像帧的稳定跟踪特征超过 30 个（程序中设定是 *15* 对，这里概念不是很清楚），且有效视差超过 20 对，则基于五点法恢复两帧之间的位姿。随后基于五点法恢复的位姿，对这两帧中跟踪到的特征点进行三角化，再基于三角化后的特征点利用 PnP 算法估计所有帧位姿，最终基于全局 BA 算法最小化所有特征点的总重投影误差。(这里原文说明的流程并不清楚，究竟是哪些帧之间的位姿，哪一帧向哪一帧的重投影误差，需要进一步参考程序。）由于纯视觉系统缺乏有关世界坐标系的先验信息，这里假设第一帧的相机坐标系（此处是否指的是窗口内的第一帧?）为参考坐标系，所有的相机位姿 $\left(\overline{\boldsymbol{p}}_{c_k}^{c_0}, \boldsymbol{q}_{c_k}^{c_0}\right)$ 和特征点位置均表示在第一帧坐标系下。记相机和 IMU 之间的外参为 $\left(\boldsymbol{p}_c^b, \boldsymbol{q}_c^b\right)$，则 IMU 坐标系在 c_0 下的位姿为：

$$\begin{aligned} \boldsymbol{q}_{b_k}^{c_0} &= \boldsymbol{q}_b^c \otimes \boldsymbol{q}_{c_k}^{c_0} \\ s\overline{\boldsymbol{p}}_{b_k}^{c_0} &= s\overline{p}_{c_k}^{c_0} - \boldsymbol{p}_b^c \end{aligned} \tag{3-6}$$

3.2.2 程序解析

滑窗内的纯视觉 SfM 包含在 GlobalSFM 类中，对应头文件和源文件为 vins_estimator/initial 文件夹下的 initial_sfm.cpp 和 initial_sfm.h，该类只有唯一的一个公共借口函数 constructor()，对应的算法流程如下：

(1) 选取包含 k 个图像帧的滑动窗口，将窗口内的第 1 帧图像对应的位姿设为基准位姿，记该图像的序号为 l；利用对极几何约束（五点法或八点法）恢复第 k 帧的相对位姿；

(2) 基于第 1 帧和第 k 帧的位姿三角化部分特征点的空间位置信息；

(3) 从已三角化的特征点集合中选取能被第 2 帧图像看到的特征点，基于 PnP 求解第 2 帧的相对位姿；

(4) 基于第一帧和第 k 帧的位姿三角化部分特征点的空间位置信息，第 2 步中已经三角化的特征点会被直接跳过；

(5) 重复上述步骤直到第 $k-1$ 帧，此时滑窗内所有图像帧相对于第 1 帧的相对位姿均已获得。

(6) 依次基于第 1 帧位姿和第 i 帧位姿三角化前序过程中剩余的特征点；

(7) 基于现有的特征点空间信息，利用 PnP 求解窗口内第 $l-1$ 帧图像的相对位姿；

(8) 基于第 l 帧和第 $l-1$ 帧位姿三角化部分特征点的空间信息；

(9) 重复 7、8 两步直到系统起始时刻的第 0 帧图像；

(10) 至此，所有图像帧的相对位姿均解算完毕，对于此时仍未三角化的特征点，选择其第一次和最后一次被观测到的图像帧，进行三角化。

(11) 使用 Ceres 对现有所有图像帧的位姿和空间点的三维坐标进行 BA 优化。

3.3 惯性视觉对准

惯性视觉对准的基本思想是将 IMU 的与积分值与视觉 SfM 获得了待尺度的数值进行匹配。

3.3.1 陀螺零偏标定

考虑窗口内的连续两帧 $\boldsymbol{b}_k$ 和 $\boldsymbol{b}_{k+1}$，我们已经通过纯视觉 SfM 计算出了对应的旋转 $\boldsymbol{q}_{b_k}^{c_0}$ 和 $\boldsymbol{q}_{b_{k+1}}^{c_0}$，同时 IMU 预积分也提供了 k 帧到 $k+1$ 帧的相对姿态约束 $\boldsymbol{\gamma}_{k+1}^{b_k}$，通过最小化两者之间的误差来实现 IMU 和视觉量测的对准：

$$\min_{\delta \boldsymbol{b}_\omega} \sum_{k \in \mathcal{B}} \| \boldsymbol{q}_{b_{k+1}}^{c_0} \otimes \boldsymbol{q}_{b_k}^{c_0 -1} \otimes \boldsymbol{\gamma}_{k+1}^{b_k} \|^2 \tag{3-7}$$

将式 2-4中的 IMU 预积分关于陀螺零偏的线性形式带入，则可通过非线性优化方式得到对陀螺零偏的最优估计，再利用新获得的陀螺零偏更新 IMU 的预积分项。

3.3.2 速度矢量、重力矢量及几何尺度初始化

A KLT 稀疏光流跟踪算法

B 基于四元数微分的 IMU 预积分模型

B.1 IMU 噪声与零偏模型

IMU 的量测模型可表示为:

$$\begin{cases} \hat{\boldsymbol{a}}^b = \boldsymbol{a}^b + \boldsymbol{b}_a + \boldsymbol{R}_w^b \boldsymbol{g}^w + \boldsymbol{n}_a \\ \hat{\boldsymbol{\omega}}^b = \boldsymbol{\omega}^b + \boldsymbol{b}_\omega + \boldsymbol{n}_\omega \end{cases} \tag{B-8}$$

式中，$\hat{\boldsymbol{a}}^b$ 和 $\hat{\boldsymbol{\omega}}^b$ 分别表示 IMU 的加速度和角速度测量信息；$\boldsymbol{a}^b$ 和 $\boldsymbol{\omega}^b$ 分别为载体真实的加速度和角速度；$\boldsymbol{b}_a$ 和 $\boldsymbol{b}_\omega$ 分别为加速度计和陀螺仪零偏，建模为随机游走过程:

$$\begin{cases} \boldsymbol{n}_{b_a} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\sigma}_{b_a}^2\right), \boldsymbol{n}_{b_\omega} \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\sigma}_{b_\omega}^2\right) \\ \dot{\boldsymbol{b}_a} = \boldsymbol{n}_{b_a}, \dot{\boldsymbol{b}_\omega} = \boldsymbol{n}_{b_\omega} \end{cases} \tag{B-9}$$

$\boldsymbol{R}_w^b$ 为世界坐标系到本体坐标系的坐标旋转矩阵；$\boldsymbol{g}^w$ 为世界坐标系下的重力加速度矢量；$\boldsymbol{n}_a \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\sigma}_a^2\right)$ 和 $\boldsymbol{n}_\omega \sim \left(\boldsymbol{0}, \boldsymbol{\sigma}_\omega^2\right)$ 为器件的高斯白噪声。

B.2 四元数左乘与右乘

假设一个四元数 $\boldsymbol{q}_a = \begin{bmatrix} w_a & x_a & y_a & z_a \end{bmatrix}^T = \begin{bmatrix} w_a & \boldsymbol{v}_a \end{bmatrix}^T$，左乘和右乘的矩阵形式分别为:

$$\begin{aligned} L\left(\boldsymbol{q}_a\right) &= \begin{bmatrix} w_a & -x_a & -y_a & -z_a \\ x_a & w_a & -z_a & y_a \\ y_a & z_a & w_a & -x_a \\ z_a & -y_a & x_a & w_a \end{bmatrix} = w_a \boldsymbol{I} + \begin{bmatrix} 0 & -\boldsymbol{v}_a^T \\ \boldsymbol{v}_a & \boldsymbol{v}_a^\wedge \end{bmatrix} \\ R\left(\boldsymbol{q}_a\right) &= \begin{bmatrix} w_a & -x_a & -y_a & -z_a \\ x_a & w_a & z_a & -y_a \\ y_a & -z_a & w_a & x_a \\ z_a & y_a & -x_a & w_a \end{bmatrix} = w_a \boldsymbol{I} + \begin{bmatrix} 0 & -\boldsymbol{v}_a^T \\ \boldsymbol{v}_a & -\boldsymbol{v}_a^\wedge \end{bmatrix} \end{aligned} \tag{B-10}$$

若四元数的实部位于最后，即 $\boldsymbol{q}_a = \begin{bmatrix} x_a & y_a & z_a & w_a \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{v}_a & w_a \end{bmatrix}^T$，则对应的左乘和右乘矩阵为：

$$\begin{aligned} L\left(\boldsymbol{q}_a\right) &= \begin{bmatrix} w_a & -z_a & y_a & x_a \\ z_a & w_a & -x_a & y_a \\ -y_a & x_a & w_a & z_a \\ -x_a & -y_a & -z_a & w_a \end{bmatrix} = w_a \boldsymbol{I} + \begin{bmatrix} \boldsymbol{v}_a^{\wedge} & \boldsymbol{v}_a \\ -\boldsymbol{v}_a^T & 0 \end{bmatrix} \\ R\left(\boldsymbol{q}_a\right) &= \begin{bmatrix} w_a & z_a & -y_a & x_a \\ -z_a & w_a & x_a & y_a \\ y_a & -x_a & w_a & z_a \\ -x_a & -y_a & -z_a & w_a \end{bmatrix} = w_a \boldsymbol{I} + \begin{bmatrix} -\boldsymbol{v}_a^{\wedge} & \boldsymbol{v}_a \\ -\boldsymbol{v}_a^T & 0 \end{bmatrix} \end{aligned} \tag{B-11}$$

B.3 四元数微分

假设世界坐标系旋转到本体坐标系的旋转轴为 $\boldsymbol{n}$，旋转角为 θ，则对应的旋转四元数为 $\boldsymbol{q} = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)\boldsymbol{n}$，该单位四元数对时间的导数为：

$$\frac{d\boldsymbol{q}}{dt} = -\frac{1}{2}\sin\left(\frac{\theta}{2}\right)\cdot\frac{d\theta}{dt} + \frac{d\boldsymbol{n}}{dt}\cdot\sin\left(\frac{\theta}{2}\right) + \boldsymbol{n}\cdot\frac{1}{2}\cos\left(\frac{\theta}{2}\right)\cdot\frac{d\theta}{dt} \tag{B-12}$$

式中 $-1 = \boldsymbol{n}\cdot\boldsymbol{n}$，$\frac{d\boldsymbol{n}}{dt} = 0$，$\frac{d\theta}{dt} = \omega_w^w b$，$\omega_w^w b$ 为世界坐标系下世界坐标系到本体系的旋转角速度。对上式进一步化简有：

$$\begin{aligned} \frac{d\boldsymbol{q}}{dt} &= \frac{1}{2}\boldsymbol{n}\omega_{wb}^w\left(\cos\left(\frac{\theta}{2}\right) + \boldsymbol{n}\sin\left(\frac{\theta}{2}\right)\right) \\ &= \frac{1}{2}\boldsymbol{\omega}_{wb}^w\boldsymbol{q} \end{aligned} \tag{B-13}$$

由于陀螺测量得到的角速度为本体系下的角速度 $\boldsymbol{\omega}_{wb}^b$，和世界坐标系下角速度的关系*可由四元数表示 $\boldsymbol{q}\boldsymbol{\omega}_{wb}^b\boldsymbol{q}^* = \boldsymbol{\omega}_{wb}^w$，带入上式有：

$$\frac{d\boldsymbol{q}}{dt} = \frac{1}{2}\boldsymbol{q}\boldsymbol{\omega}_{wb}^b\boldsymbol{q}^*\boldsymbol{q} \tag{B-14}$$

利用单位四元数的性质 $\boldsymbol{q}\boldsymbol{q}^{-1} = \boldsymbol{q}\boldsymbol{q}^* = \boldsymbol{I}$ 得到四元数微分和陀螺测量角速度之间的关系为：

$$\frac{d\boldsymbol{q}}{dt} = \frac{1}{2}\boldsymbol{q}\boldsymbol{\omega}_{wb}^b \tag{B-15}$$

*假设四元数 $\boldsymbol{q}$ 表示世界坐标系到本体坐标系的旋转，即 $f^b = \boldsymbol{q}f^w\boldsymbol{q}^*$。而此处矢量本身并未旋转，只因坐标系旋转引起了坐标的变化，相当于坐标系旋转的逆过程，因此有 $\boldsymbol{w}_{wb}^b = \boldsymbol{q}^*\boldsymbol{w}_{wb}^w\boldsymbol{q}$。

上式写成矩阵形式为：

$$\begin{aligned}\frac{d\boldsymbol{q}}{dt} &= \frac{1}{2}\begin{bmatrix}0 & -\omega_x & -\omega_y & -\omega_z\\ \omega_x & 0 & \omega_z & -\omega_y\\ \omega_y & -\omega_z & 0 & \omega_x\\ \omega_z & \omega_y & -\omega_x & 0\end{bmatrix}\boldsymbol{q}\\ &= \frac{1}{2}\begin{bmatrix}0 & -\boldsymbol{\omega}_{wb}^{b\ T}\\ \boldsymbol{\omega}_{wb}^{b} & -\boldsymbol{\omega}_{wb}^{b\ \wedge}\end{bmatrix}\\ &= \frac{1}{2}\boldsymbol{\Omega}\left(\boldsymbol{\omega}_{wb}^{b}\right)\boldsymbol{q}\end{aligned} \tag{B-16}$$

该矩阵形式默认四元数的排列为实部在前。

此外，这里说明下四元数和坐标旋转矩阵之间的关系，假设 w 系旋转到 b 系对应的旋转四元数为 $\boldsymbol{q}_w^b$，坐标旋转矩阵为 $\boldsymbol{R}_w^b$，则对于任意一个矢量 $\boldsymbol{a}$ 有 $\boldsymbol{R}_w^b\boldsymbol{a} = \boldsymbol{q}_w^{b\ *}\boldsymbol{a}\boldsymbol{q}_w^b$，展开得：

$$\begin{aligned}\boldsymbol{R}_w^b\boldsymbol{a} &= \begin{bmatrix}\left(q_w^2+q_x^2-q_y^2-q_z^2\right)x + 2\left(q_wq_z+q_xq_y\right)y + 2\left(q_xq_z-q_2q_y\right)z\\ 2\left(q_xq_y-q_wq_z\right)x + \left(q_w^2-q_x^2+q_y^2-q_z^2\right)y + 2\left(q_wq_x+q_yq_z\right)z\\ 2\left(q_wq_y-q_xq_z\right)x + 2\left(q_yq_z-q_wq_x\right)y + \left(q_w^2-q_x^2-q_y^2+q_z^2\right)z\end{bmatrix}\\ &= \begin{bmatrix}\left(q_w^2+q_x^2-q_y^2-q_z^2\right) & 2\left(q_wq_z+q_xq_y\right) & 2\left(q_xq_z-q_2q_y\right)\\ 2\left(q_xq_y-q_wq_z\right) & \left(q_w^2-q_x^2+q_y^2-q_z^2\right) & 2\left(q_wq_x+q_yq_z\right)\\ 2\left(q_wq_y-q_xq_z\right) & 2\left(q_yq_z-q_wq_x\right) & \left(q_w^2-q_x^2-q_y^2+q_z^2\right)\end{bmatrix}\begin{bmatrix}x\\ y\\ z\end{bmatrix}\end{aligned} \tag{B-17}$$

B.4 四元数微分的毕卡算法

式 B-15所表示的四元数微分方程可采用毕卡级数求解，这里省略详细的推导过程，直接给出四元数的毕卡积分解：

$$\boldsymbol{q}_{t_{k+1}} = e^{\frac{1}{2}\int_{t_k}^{t_{k+1}}\boldsymbol{\Omega}\left(\boldsymbol{\omega}_{\boldsymbol{wb}}^{\boldsymbol{b}}\right)dt}\boldsymbol{q}_{t_k} \tag{B-18}$$

记 $\Delta\boldsymbol{\Theta} = \int_{t_k}^{t_{k+1}}\boldsymbol{\Omega}\left(\boldsymbol{\omega}_{\boldsymbol{wb}}^{\boldsymbol{b}}\right)dt$，则：

$$\begin{aligned}\Delta\Theta &= \int_k^{k+1}\begin{bmatrix}0 & -\omega_x & -\omega_y & -\omega_z\\ \omega_x & 0 & \omega_z & -\omega_y\\ \omega_y & -\omega_z & 0 & \omega_x\\ \omega_z & \omega_y & -\omega_x & 0\end{bmatrix}dt\\ &\approx \begin{bmatrix}0 & -\Delta\theta_x & -\Delta\theta_y & -\Delta\theta_z\\ \Delta\theta_x & 0 & \Delta\theta_z & -\Delta\theta_y\\ \Delta\theta_y & -\Delta\theta_z & 0 & \Delta\theta_x\\ \Delta\theta_z & \Delta\theta_y & -\Delta\theta_x & 0\end{bmatrix}\end{aligned} \tag{B-19}$$

$\Delta\boldsymbol{\theta}=\begin{bmatrix}\Delta\theta_x & \Delta\theta_y & \Delta\theta_z\end{bmatrix}$ 称为采样时间 $[t_k,t_{k+1}]$ 内的角增量。将上式带入式 B-18中并对其进行泰勒展开有：

$$\begin{aligned}\boldsymbol{q}_{t_{k+1}}&=e^{\frac{1}{2}\Delta\boldsymbol{\Theta}}\boldsymbol{q}_{t_k}\\&=\left(\boldsymbol{I}+\frac{\frac{1}{2}\Delta\boldsymbol{\Theta}}{1!}+\frac{\left(\frac{1}{2}\Delta\boldsymbol{\Theta}\right)^2}{2!}+\frac{\left(\frac{1}{2}\Delta\boldsymbol{\Theta}\right)^3}{3!}+\frac{\left(\frac{1}{2}\Delta\boldsymbol{\Theta}\right)^4}{4!}+o\left(\frac{1}{2}\Delta\boldsymbol{\Theta}\right)\right)\boldsymbol{q}_{t_k}\end{aligned}\tag{B-20}$$

上式中，$o\left(\frac{1}{2}\Delta\boldsymbol{\Theta}\right)$ 为高阶小量。$\Delta\boldsymbol{\Theta}$ 的幂函数存在一定的规律：

$$\begin{aligned}\Delta\boldsymbol{\Theta}^2&=-\Delta\Theta^2\boldsymbol{I}\\\Delta\boldsymbol{\Theta}^3&=-\Delta\Theta^2\Delta\boldsymbol{\Theta}\\\Delta\boldsymbol{\Theta}^4&=-\Delta\Theta^4\boldsymbol{I}\\\Delta\boldsymbol{\Theta}^5&=\Delta\Theta^4\Delta\boldsymbol{\Theta}\\&\cdots\end{aligned}\tag{B-21}$$

将上式带入式 B-20并合并包含和不包含 $\Delta\boldsymbol{\Theta}$ 的同类项有：

$$\begin{aligned}\boldsymbol{q}_{t_{k+1}}&=\left\{\boldsymbol{I}\left(1-\frac{\left(\frac{\Delta\theta}{2}\right)^2}{2!}+\frac{\left(\frac{\Delta\theta}{2}\right)^4}{4!}-\frac{\left(\frac{\Delta\theta}{2}\right)^6}{6!}+\cdots\right)+\frac{\Delta\boldsymbol{\Theta}}{2}\left(\frac{\left(\frac{\Delta\theta}{2}\right)}{1!}-\frac{\left(\frac{\Delta\theta}{2}\right)^3}{3!}+\frac{\left(\frac{\Delta\theta}{2}\right)^5}{5!}-\cdots\right)\frac{2}{\Delta\theta}\right\}\boldsymbol{q}_{t_k}\\&=\left(\boldsymbol{I}\cos\frac{\Delta\theta}{2}+\Delta\boldsymbol{\Theta}\frac{\sin\frac{\Delta\theta}{2}}{\Delta\theta}\right)\boldsymbol{q}_{t_k}\end{aligned}\tag{B-22}$$

上式即为完整形式的四元数角增量毕卡算法。但实际应用中，为了避免计算三角函数，通常直接取近似解：

(1) 一阶近似毕卡算法：

$$\boldsymbol{q}_{t_k+1}=\left(\boldsymbol{I}+\frac{\Delta\boldsymbol{\Theta}}{2}\right)\boldsymbol{q}_{t_k}\tag{B-23}$$

(2) 二阶近似毕卡算法：

$$\boldsymbol{q}_{t_k+1}=\left[\boldsymbol{I}\left(1-\frac{\Delta\theta^2}{8}\right)+\frac{\Delta\boldsymbol{\Theta}}{2}\right]\boldsymbol{q}_{t_k}\tag{B-24}$$

(3) 三阶近似毕卡算法：

$$\boldsymbol{q}_{t_k+1}=\left[\boldsymbol{I}\left(1-\frac{\Delta\theta^2}{8}\right)+\left(\frac{1}{2}-\frac{\Delta\theta^2}{48}\right)\Delta\boldsymbol{\Theta}\right]\boldsymbol{q}_{t_k}\tag{B-25}$$

(4) 四阶近似毕卡算法：

$$\boldsymbol{q}_{t_k+1}=\left[\boldsymbol{I}\left(1-\frac{\Delta\theta^2}{8}+\frac{\Delta\theta^4}{384}\right)+\left(\frac{1}{2}-\frac{\Delta\theta^2}{48}\right)\Delta\boldsymbol{\Theta}\right]\boldsymbol{q}_{t_k}\tag{B-26}$$

其中，将一阶毕卡算法写成四元数乘法形式为:

$$\boldsymbol{q}_{t_{k+1}} = \boldsymbol{q}_{t_k} \otimes \begin{bmatrix} 1 \\ \frac{\Delta\theta_x}{2} \\ \frac{\Delta\theta_y}{2} \\ \frac{\Delta\theta_z}{2} \end{bmatrix} \tag{B-27}$$

进一步离散化有:

$$\boldsymbol{q}_{t_{k+1}} = \boldsymbol{q}_{t_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\boldsymbol{\omega}_{wb}^{b}\delta t \end{bmatrix} \tag{B-28}$$

B.5 基于四元数的 IMU 预积分

B.5.1 连续时间形式

载体的位置、速度和姿态四元数在时间区间 $[t_k, t_{k+1}]$ 内的传播方程为:

$$\begin{cases} \boldsymbol{p}_{k+1}^{w} = \boldsymbol{p}_{k}^{w} + \boldsymbol{v}_{k}^{w}\Delta t_k + \iint_{t_i\in[t_k,t_{k+1}]} \left(\boldsymbol{R}_{b_{t_i}}^{w}\left(\hat{\boldsymbol{a}}_{t_i} - \boldsymbol{b}_a - \boldsymbol{n}_a\right) - \boldsymbol{g}^{w}\right) dt^2 \\ \boldsymbol{v}_{k+1}^{w} = \boldsymbol{v}_{k}^{w} + \int_{t_i\in[t_k,t_{k+1}]} \left(\boldsymbol{R}_{b_{t_i}}^{w}\left(\hat{\boldsymbol{a}}_{t_i} - \boldsymbol{b}_a - \boldsymbol{n}_a\right) - \boldsymbol{g}^{w}\right) dt \\ \boldsymbol{q}_{w}^{b_{k+1}} = \boldsymbol{q}_{w}^{b_k} \otimes \int_{t_i\in[t_k,t_{k+1}]} \frac{1}{2}\boldsymbol{\Omega}\left(\hat{\boldsymbol{\omega}}_{wb}^{w} - \boldsymbol{b}_\omega - \boldsymbol{n}_\omega\right)\boldsymbol{q}_{k}^{w} dt \end{cases} \tag{B-29}$$

式中 $\boldsymbol{q}_{w}^{b_{k+1}}$ 为世界坐标系到本体坐标系的旋转四元数，$\otimes$ 表示四元数乘法，$\boldsymbol{\Omega}(\boldsymbol{\omega})$ 为上一小节中由 $\boldsymbol{\omega}$ 组成的齐次矩阵，$\boldsymbol{b}_k$ 和 $\boldsymbol{b}_{k+1}$ 分别为第 k 帧和第 $k+1$ 帧对应的体坐标系，$\boldsymbol{b}_{t_i}$ 为 t_i 时刻的体坐标系。

可以看出，IMU 的由第 k 帧向第 $k+1$ 帧的状态传播需要第 k 帧的旋转、位置和速度信息，这使得后端优化时，每当 k 帧位姿优化完后均需要重新计算 $k+1$ 的相关变量；而若将 B-29中右侧的积分项表示在 k 帧对应的本体系下，则该项不包含任何需要后端优化的位姿变量而只依赖于 IMU 的量测信息，更加便于后端优化处理。因此，增量形式的 IMU 状态传播方程为:

$$\begin{cases} \boldsymbol{R}_{w}^{b_k}\boldsymbol{p}_{k+1}^{w} = \boldsymbol{R}_{w}^{b_k}\left(\boldsymbol{p}_{k}^{w} + \boldsymbol{v}_{k}^{w}\Delta t_k - \frac{1}{2}\boldsymbol{g}^{w}\Delta t_k^2\right) + \hat{\boldsymbol{\alpha}}_{k+1}^{b_k} \\ \boldsymbol{R}_{w}^{b_k}\boldsymbol{v}_{k+1}^{w} = \boldsymbol{R}_{w}^{b_k}\left(\boldsymbol{v}_{k}^{w} - \boldsymbol{g}^{w}\Delta t_k\right) + \hat{\boldsymbol{\beta}}_{k+1}^{b_k} \\ \boldsymbol{q}_{b_k}^{w} \otimes \boldsymbol{q}_{w}^{b_{k+1}} = \hat{\boldsymbol{\gamma}}_{k+1}^{b_k} \end{cases} \tag{B-30}$$

式中，$\boldsymbol{\alpha}_{k+1}^{b_k}$、$\boldsymbol{\beta}_{k+1}^{b_k}$ 和 $\boldsymbol{\gamma}_{\boldsymbol{k+1}}^{\boldsymbol{b_k}}$ 分别为 b_k 坐标系下第 k 和 $k+1$ 帧之间 IMU 的预积分项：

$$\begin{cases} \hat{\boldsymbol{\alpha}}_{k+1}^{b_k} = \iint_{t_i \in [t_k, t_{k+1}]} \left(\boldsymbol{R}_{b_{t_i}}^{w} \left(\hat{\boldsymbol{a}}_{t_i} - \boldsymbol{b}_a - \boldsymbol{n}_a \right) \right) dt^2 \\ \hat{\boldsymbol{\beta}}_{k+1}^{b_k} = \int_{t_i \in [t_k, t_{k+1}]} \left(\boldsymbol{R}_{b_{t_i}}^{w} \left(\hat{\boldsymbol{a}}_{t_i} - \boldsymbol{b}_a - \boldsymbol{n}_a \right) \right) dt \\ \hat{\boldsymbol{\gamma}}_{k+1}^{b_k} = \int_{t_i \in [t_k, t_{k+1}]} \frac{1}{2} \boldsymbol{\Omega} \left(\hat{\boldsymbol{\omega}}_{wb}^{w} - \boldsymbol{b}_\omega - \boldsymbol{n}_\omega \right) \boldsymbol{\gamma}_k^{t_i} dt \end{cases} \tag{B-31}$$

可以看出，B-31中的 IMU 预积分项只和实际 IMU 期间量测以及 IMU 零偏有关。当后端优化的 IMU 零偏变化时，若变化较小，则使用 IMU 预积分项关于零偏的一阶近似更新预积分项；否则，则按照式 B-31重新积分 IMU 得到预积分项。IMU 预积分关于零偏的线性化更新表达式可以表示为：

$$\begin{cases} \hat{\boldsymbol{\alpha}}_k^{k+1} \approx \hat{\boldsymbol{\alpha}}_k^{k+1} + \boldsymbol{J}_{\boldsymbol{b}_a}^{\alpha} \delta \boldsymbol{b}_a + \boldsymbol{J}_{\boldsymbol{b}_\omega}^{\alpha} \delta \boldsymbol{b}_\omega \\ \hat{\boldsymbol{\beta}}_k^{k+1} \approx \hat{\boldsymbol{\beta}}_k^{k+1} + \boldsymbol{J}_{\boldsymbol{b}_a}^{\beta} \delta \boldsymbol{b}_a + \boldsymbol{J}_{\boldsymbol{b}_\omega}^{\beta} \delta \boldsymbol{b}_\omega \\ \hat{\boldsymbol{\gamma}}_k^{k+1} = \hat{\boldsymbol{\gamma}}_k^{k+1} \otimes \boldsymbol{J}_{\boldsymbol{b}_\omega}^{\gamma} \delta \boldsymbol{b}_\omega \end{cases} \tag{B-32}$$

B.5.2 离散时间形式

为了便于实际程序的执行，需要将上述连续形式的预积分方程改写为离散形式，由于噪声均值为零，预积分中可以忽略不计，因此两个 IMU 量测时刻 i 和 $i+1$ 之间离散形式的 IMU 预积分方程为：

$$\begin{cases} \boldsymbol{p}_{i+1}^{w} = \boldsymbol{p}_i^{w} + \boldsymbol{v}_i^{w} \Delta t_i + \frac{1}{2} \overline{\boldsymbol{a}}_{t_i} \delta t^2 \\ \boldsymbol{v}_{i+1}^{w} = \boldsymbol{v}_i^{w} + \overline{\boldsymbol{a}}_{t_i} \delta t \\ \boldsymbol{q}_{w}^{b_{i+1}} = \boldsymbol{q}_{w}^{b_i} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \overline{\boldsymbol{\omega}}_{t_i} \delta t \end{bmatrix} \end{cases} \tag{B-33}$$

式中，$\overline{\boldsymbol{a}}_{t_i}$ 和 $\overline{\boldsymbol{\omega}}_{t_i}$ 分别表示离散区间内的离散加速度，具体表达形式由离散方法决定：

(1) 欧拉法：

$$\begin{cases} \overline{\boldsymbol{a}}_{t_i} = \boldsymbol{R}_{b_{t_i}}^{w} \left(\hat{\boldsymbol{a}}_{t_i} - \boldsymbol{b}_a \right) - \boldsymbol{g}^{w} \\ \overline{\boldsymbol{\omega}}_{t_i} = \hat{\boldsymbol{\omega}}_{t_i} - \boldsymbol{b}_\omega \end{cases} \tag{B-34}$$

(2) 中值法：

$$\begin{cases} \overline{\boldsymbol{a}}_{t_i} = \frac{1}{2} \left[\boldsymbol{R}_{b_{t_i}}^{w} \left(\hat{\boldsymbol{a}}_{t_i} - \boldsymbol{b}_a \right) + \boldsymbol{R}_{b_{t_{i+1}}}^{w} \left(\hat{\boldsymbol{a}}_{t_{i+1}} - \boldsymbol{b}_a \right) \right] - \boldsymbol{g}^{w} \\ \overline{\boldsymbol{\omega}}_{t_i} = \frac{1}{2} \left(\hat{\boldsymbol{\omega}}_{t_i} + \hat{\boldsymbol{\omega}}_{t_{i+1}} \right) - \boldsymbol{b}_\omega \end{cases} \tag{B-35}$$

同理，可得到离散的增量形式 IMU 预积分方程为：

$$\begin{cases} \hat{\boldsymbol{\alpha}}_{i+1}^{b_k} = \hat{\boldsymbol{\alpha}}_i^{b_k} + \hat{\boldsymbol{\beta}}_i^{b_k}\delta t + \frac{1}{2}\overline{\boldsymbol{a}}_i\delta t^2 \\ \hat{\boldsymbol{\beta}}_{i+1}^{b_k} = \hat{\boldsymbol{\beta}}_i^{b_k} + \overline{\boldsymbol{a}}_i\delta t \\ \hat{\boldsymbol{\gamma}}_{i+1}^{b_k} = \hat{\boldsymbol{\gamma}}_i^{b_k} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\overline{\boldsymbol{\omega}}_i\delta t \end{bmatrix} \end{cases} \tag{B-36}$$

同样有欧拉法和中值法两种离散形式：

(1) 欧拉法：

$$\begin{cases} \overline{\boldsymbol{a}}_i = \boldsymbol{R}\left(\hat{\boldsymbol{\gamma}}_i^{b_k}\right)(\hat{\boldsymbol{a}}_i - \boldsymbol{b}_a) \\ \overline{\boldsymbol{\omega}}_i = \hat{\boldsymbol{\omega}}_i - \boldsymbol{b}_\omega \end{cases} \tag{B-37}$$

(2) 中值法：

$$\begin{cases} \overline{\boldsymbol{a}}_i = \frac{1}{2}\left[\boldsymbol{R}\left(\hat{\boldsymbol{\gamma}}_i^{b_k}\right)(\hat{\boldsymbol{a}}_i - \boldsymbol{b}_a) + \boldsymbol{R}\left(\hat{\boldsymbol{\gamma}}_{i+1}^{b_k}\right)(\hat{\boldsymbol{a}}_{i+1} - \boldsymbol{b}_a)\right] \\ \overline{\boldsymbol{\omega}}_i = \frac{1}{2}(\hat{\boldsymbol{\omega}}_i + \hat{\boldsymbol{\omega}}_{i+1}) - \boldsymbol{b}_\omega \end{cases} \tag{B-38}$$

B.6 IMU 预积分误差动力学方程与协方差矩阵

在 VINS 的优化中，由于采用马氏距离构建二乘函数，需要对状态的协方差矩阵进行更新，而协方差矩阵的一步预测依赖于状态的误差动力学方程，因此这里我们首先就 IMU 预积分项的误差动力学方程进行推导。涉及的状态量包括位置增量 $\boldsymbol{\alpha}_{k+1}^{b_k}$、速度增量 $\boldsymbol{\beta}_{k+1}^{b_k}$、姿态增量 $\boldsymbol{\theta}_{k+1}^{b_k}$、加速度计零偏 $\boldsymbol{b}_a$ 和陀螺仪零偏 $\boldsymbol{b}_\omega$，其中由于陀螺器件测量为角增量，这里使用角增量 $\boldsymbol{\theta}_{k+1}^{b_k}$ 代替四元数增量 $\boldsymbol{\gamma}_{k+1}^{b_k}$。

B.6.1 连续时间形式

由于我们需要求解的是上述各增量的误差动力学方程，因此系统状态空间矢量可记为*。

$$\delta\boldsymbol{x} = \begin{bmatrix} \delta\boldsymbol{\alpha}_t^{b_k} & \delta\boldsymbol{\beta}_t^{b_k} & \delta\boldsymbol{\theta}_t^{b_k} & \delta\boldsymbol{b}_a & \delta\boldsymbol{b}_\omega & \boldsymbol{n}_a & \boldsymbol{n}_\omega & \boldsymbol{n}_{\boldsymbol{b}_a} & \boldsymbol{n}_{\boldsymbol{b}_\omega} \end{bmatrix} \tag{B-39}$$

由于导数运算的顺序可以交换，即 $\delta\dot{\boldsymbol{a}} = \delta\dot{\boldsymbol{a}}$，我们可以先求当前变量的时间导数，再求其对各状态量的偏导数。

(1) **位置增量误差** $\delta\boldsymbol{\alpha}_t^{b_k}$

位置增量的时间导数为 $\dot{\boldsymbol{\alpha}}_t^{b_k} = \boldsymbol{\beta}_t^{b_k}$，显然，除了偏导数 $\delta\dot{\boldsymbol{\alpha}}_t^{b_k}/\delta\boldsymbol{\beta}_t^{b_k} = \boldsymbol{I}$ 外，位置增量的导数对其余状态增量的导数为零，即其余状态增量的变化对位置增量误差的导数没有贡献.

*由于增量表示的是 $k+1$ 和 k 之间的相对值，这里为了书写方便，下标使用 k 而非前面 $k+1$，同时改用下标 t 表示 IMU 测量时刻，以区别图像帧时刻 k，但两者本质上是同一个量。

(2) **速度增量误差** $\delta\boldsymbol{\beta}_t^{b_k}$

速度增量的时间导数为 $\dot{\boldsymbol{\beta}}_t^{b_k}=\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t-\boldsymbol{b}_a-\boldsymbol{n}_a\right)$，可以看出非零偏导数共有三项，分别为：

$$\begin{aligned}\frac{\delta\boldsymbol{\beta}_t^{b_k}}{\delta\boldsymbol{\theta}_k^{b_k}}&=-\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t-\boldsymbol{b}_a-\boldsymbol{n}_a\right)\\&\approx-\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t-\boldsymbol{b}_a\right)\\\frac{\delta\boldsymbol{\beta}_t^{b_k}}{\boldsymbol{b}_a}&=-\boldsymbol{R}_{b_t}^{b_k}\\\frac{\delta\boldsymbol{\beta}_t^{b_k}}{\boldsymbol{n}_a}&=-\boldsymbol{R}_{b_t}^{b_k}\end{aligned}\tag{B-40}$$

其中，由于状态递推的本质是随机变量均值的传播，而 $\boldsymbol{n}_a$ 是零均值的高斯白噪声，可作为零值处理。

(3) **角增量误差** $\delta\boldsymbol{\theta}_t^{b_k}$

由于角增量的运算中涉及四元数的乘法，使用导数微分定义求解会比较复杂，这里我们换一种思路，假设无任何偏差的四元数时间导数和带各项扰动的四元数时间导数分别为 $\dot{\boldsymbol{q}}_n$ 和 $\dot{\boldsymbol{q}}_t$，则：

$$\begin{aligned}\dot{\boldsymbol{q}}_t&=\frac{1}{2}\underbrace{\boldsymbol{q}\otimes\delta\boldsymbol{q}}_{\delta\boldsymbol{\theta}}\otimes\begin{bmatrix}0\\\hat{\boldsymbol{\omega}}-\boldsymbol{b}_\omega-\boldsymbol{n}_\omega-\delta\boldsymbol{b}_\omega\end{bmatrix}\\\dot{\boldsymbol{q}}_n&=\frac{1}{2}\boldsymbol{q}\otimes\begin{bmatrix}0\\\hat{\boldsymbol{\omega}}-\boldsymbol{b}_\omega\end{bmatrix}\end{aligned}\tag{B-41}$$

同时，$\dot{\boldsymbol{q}}_t$ 又可表示为：

$$\begin{aligned}\dot{\boldsymbol{q}}_t&=\frac{\partial\left(\boldsymbol{q}\otimes\delta\boldsymbol{q}\right)}{\partial t}=\dot{\boldsymbol{q}}\otimes\delta\boldsymbol{q}+\boldsymbol{q}\otimes\dot{\delta\boldsymbol{q}}\\&=\frac{1}{2}\boldsymbol{q}\otimes\begin{bmatrix}0\\\boldsymbol{\omega}-\boldsymbol{b}_\omega\end{bmatrix}\otimes\delta\boldsymbol{q}+\boldsymbol{q}\otimes\dot{\delta\boldsymbol{q}}\end{aligned}\tag{B-42}$$

令上述两式相等可得到：

$$\begin{aligned}&\frac{1}{2}\boldsymbol{q}\otimes\delta\boldsymbol{q}\otimes\begin{bmatrix}0\\\hat{\boldsymbol{\omega}}-\boldsymbol{b}_\omega-\boldsymbol{n}_\omega-\delta\boldsymbol{b}_\omega\end{bmatrix}=\frac{1}{2}\boldsymbol{q}\otimes\begin{bmatrix}0\\\boldsymbol{\omega}-\boldsymbol{b}_\omega\end{bmatrix}\otimes\delta\boldsymbol{q}+\boldsymbol{q}\otimes\dot{\delta\boldsymbol{q}}\\\Rightarrow&\frac{1}{2}\delta\boldsymbol{q}\otimes\begin{bmatrix}0\\\hat{\boldsymbol{\omega}}-\boldsymbol{b}_\omega-\boldsymbol{n}_\omega-\delta\boldsymbol{b}_\omega\end{bmatrix}=\frac{1}{2}\begin{bmatrix}0\\\boldsymbol{\omega}-\boldsymbol{b}_\omega\end{bmatrix}\otimes\delta\boldsymbol{q}+\dot{\delta\boldsymbol{q}}\\\Rightarrow&2\dot{\delta\boldsymbol{q}}=\delta\boldsymbol{q}\otimes\begin{bmatrix}0\\\hat{\boldsymbol{\omega}}-\boldsymbol{b}_\omega-\boldsymbol{n}_\omega-\delta\boldsymbol{b}_\omega\end{bmatrix}-\begin{bmatrix}0\\\boldsymbol{\omega}-\boldsymbol{b}_\omega\end{bmatrix}\otimes\delta\boldsymbol{q}\\\Rightarrow&2\dot{\delta\boldsymbol{q}}=\mathcal{R}\left(\begin{bmatrix}0\\\hat{\boldsymbol{\omega}}-\boldsymbol{b}_\omega-\boldsymbol{n}_\omega-\delta\boldsymbol{b}_\omega\end{bmatrix}\right)\delta\boldsymbol{q}-\mathcal{L}\left(\begin{bmatrix}0\\\boldsymbol{n}_\omega+\delta\boldsymbol{b}_\omega\end{bmatrix}\right)\delta\boldsymbol{q}\\\Rightarrow&2\dot{\delta\boldsymbol{q}}=\begin{bmatrix}0&\left(\boldsymbol{n}_\omega+\delta\boldsymbol{b}_\omega\right)^T\\-\left(\boldsymbol{n}_\omega-\delta\boldsymbol{b}_\omega\right)&-\left(2\hat{\boldsymbol{\omega}}-2\boldsymbol{b}_\omega-\boldsymbol{n}_\omega-\delta\boldsymbol{b}_\omega\right)^\wedge\end{bmatrix}\begin{bmatrix}1\\\frac{\delta\boldsymbol{\theta}}{2}\end{bmatrix}\end{aligned}\tag{B-43}$$

将 $\dot{\delta \boldsymbol{q}}=\begin{bmatrix}0\\ \frac{1}{2}\dot{\delta\boldsymbol{\theta}}\end{bmatrix}$ 带入上式可以并忽略零均值项 $\boldsymbol{n}_\omega$ 以及二阶小量 $\delta\boldsymbol{b}_\omega\delta\boldsymbol{\theta}$ 可以得到：

$$\begin{aligned}\begin{bmatrix}0\\ \dot{\delta\boldsymbol{\theta}}\end{bmatrix}&=\begin{bmatrix}0 & \left(\boldsymbol{n}_\omega+\delta\boldsymbol{b}_\omega\right)^T\\ -\left(\boldsymbol{n}_\omega-\delta\boldsymbol{b}_\omega\right) & -\left(2\hat{\boldsymbol{\omega}}-2\boldsymbol{b}_\omega-\boldsymbol{n}_\omega-\delta\boldsymbol{b}_\omega\right)^\wedge\end{bmatrix}\begin{bmatrix}1\\ \frac{\delta\boldsymbol{\theta}}{2}\end{bmatrix}\\ \Rightarrow \dot{\delta\boldsymbol{\theta}}&\approx-\left(\hat{\boldsymbol{\omega}}-\boldsymbol{b}_\omega\right)^\wedge\delta\boldsymbol{\theta}-\boldsymbol{n}_\omega-\delta\boldsymbol{b}_\omega\end{aligned}\tag{B-44}$$

(4) **加速度计零偏误差** $\delta\boldsymbol{b}_a$：根据 IMU 的器件模型有 $\dot{\delta\boldsymbol{b}}_a=\boldsymbol{n}_{\boldsymbol{b}_a}$。

(5) **陀螺零偏误差** $\delta\boldsymbol{b}_\omega$：同样根据 IMU 的器件模型有 $\dot{\delta\boldsymbol{b}}_\omega=\boldsymbol{n}_{\boldsymbol{b}_\omega}$。

至此，我们已经获得了所有非零均值状态量动力学方程，写成矩阵形式为：

$$\begin{aligned}\begin{bmatrix}\dot{\delta\boldsymbol{\alpha}_t^{b_k}}\\ \dot{\delta\boldsymbol{\beta}_t^{b_k}}\\ \dot{\delta\boldsymbol{\theta}_t^{b_k}}\\ \dot{\delta\boldsymbol{b}_a}\\ \dot{\delta\boldsymbol{b}_\omega}\end{bmatrix}&=\begin{bmatrix}0 & \boldsymbol{I} & 0 & 0 & 0\\ 0 & 0 & -\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t-\boldsymbol{b}_a\right)^\wedge & -\boldsymbol{R}_{b_t}^{b_k} & 0\\ 0 & 0 & -\left(\hat{\boldsymbol{\omega}}_t-\boldsymbol{b}_\omega\right)^\wedge & 0 & -\boldsymbol{I}\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{\alpha}_t^{b_k}\\ \delta\boldsymbol{\beta}_t^{b_k}\\ \delta\boldsymbol{\theta}_t^{b_k}\\ \delta\boldsymbol{b}_a\\ \delta\boldsymbol{b}_\omega\end{bmatrix}\\ &+\begin{bmatrix}0 & 0 & 0 & 0\\ -\boldsymbol{R}_{b_t}^{b_k} & 0 & 0 & 0\\ 0 & -\boldsymbol{I} & 0 & 0\\ 0 & 0 & \boldsymbol{I} & 0\\ 0 & 0 & 0 & \boldsymbol{I}\end{bmatrix}\begin{bmatrix}\boldsymbol{n}_a\\ \boldsymbol{n}_\omega\\ \boldsymbol{n}_{\boldsymbol{b}_a}\\ \boldsymbol{n}_{\boldsymbol{b}_\omega}\end{bmatrix}\\ \Rightarrow \dot{\delta\boldsymbol{x}_t}&=\boldsymbol{f}_t\delta\boldsymbol{x}_t+\boldsymbol{v}_t\boldsymbol{n}_t\end{aligned}\tag{B-45}$$

根据导数的定义，可以将上式改写为递推形式：

$$\begin{aligned}\frac{\delta\boldsymbol{x}_{t+1}-\delta\boldsymbol{x}_t}{\delta t}&=\boldsymbol{f}_t\delta\boldsymbol{x}_t+\boldsymbol{v}_t\boldsymbol{n}_t\\ \delta\boldsymbol{x}_{t+1}&=\delta\boldsymbol{x}_t+\boldsymbol{f}_t\delta\boldsymbol{x}_t\delta t+\boldsymbol{v}_t\boldsymbol{n}_t\delta t\\ &=\underbrace{\left(\boldsymbol{I}+\boldsymbol{f}_t\delta t\right)}_{\boldsymbol{F}_t}\delta\boldsymbol{x}_t+\underbrace{\boldsymbol{v}_t\delta t}_{\boldsymbol{V}_t}\boldsymbol{n}_t\end{aligned}\tag{B-46}$$

可以看出，上式是一个标准的状态空间模型，给出了随机变量均值的传播方程；同样，我们可以直接写出随机变量方差的传播方程：

$$\begin{aligned}\boldsymbol{P}_{t+1}&=\boldsymbol{F}_t\boldsymbol{P}_t\boldsymbol{F}_t^T+\boldsymbol{V}_t\boldsymbol{Q}_t\boldsymbol{V}_t^T\\ \boldsymbol{P}_0&=\boldsymbol{0}\\ \boldsymbol{Q}_0&=\begin{bmatrix}\sigma_a^2 & 0 & 0 & 0\\ 0 & \sigma_\omega^2 & 0 & 0\\ 0 & 0 & \sigma_{\boldsymbol{b}_a}^2 & 0\\ 0 & 0 & 0 & \sigma_{\boldsymbol{b}_\omega}^2\end{bmatrix}\end{aligned}\tag{B-47}$$

误差系数自身的雅克比矩阵为：

$$\begin{aligned} \boldsymbol{J}_{k+1} &= \prod_{t\in[k,k+1]} \boldsymbol{F}_t \boldsymbol{J}_k \\ \boldsymbol{J}_k &= \boldsymbol{I} \end{aligned} \tag{B-48}$$

注意这里的下标为图像帧 k 而非 IMU 量测 t。

B.6.2 离散时间形式

同样，这里我们基于连续时间形式的误差动力学给出离散形式的误差递推方程，由于程序中采用的是中值积分，这里仅推导中值积分下的离散误差方程

(1) **角增量误差** $\delta\boldsymbol{\alpha}_t^{b_k}$

由于速度误差中耦合了角增量误差，这里我们先给出角增量误差的离散形式，中值点导数 $\overline{\delta\dot{\boldsymbol{\theta}}}$ 为：

$$\begin{aligned} \delta\dot{\boldsymbol{\theta}}_t &= -\left(\hat{\boldsymbol{\omega}}_t - \boldsymbol{b}_\omega\right)^\wedge \delta\boldsymbol{\theta}_t - \boldsymbol{n}_{\omega t} - \delta\boldsymbol{b}_\omega \\ \delta\dot{\boldsymbol{\theta}}_{t+1} &= -\left(\hat{\boldsymbol{\omega}}_{t+1} - \boldsymbol{b}_\omega\right)^\wedge \delta\boldsymbol{\theta}_t - \boldsymbol{n}_{\omega t+1} - \delta\boldsymbol{b}_\omega \\ \overline{\delta\dot{\boldsymbol{\theta}}} &= \frac{\left(\delta\dot{\boldsymbol{\theta}}_t + \delta\dot{\boldsymbol{\theta}}_{t+1}\right)}{2} \\ &= -\left(\frac{\hat{\boldsymbol{\omega}}_t + \hat{\boldsymbol{\omega}}_{t+1}}{2} - \boldsymbol{b}_\omega\right)^\wedge \delta\boldsymbol{\theta}_t - \frac{\boldsymbol{n}_{\omega t} + \boldsymbol{n}_{\omega t+1}}{2} - \delta\boldsymbol{b}_\omega \end{aligned} \tag{B-49}$$

根据导数定义展开得到：

$$\begin{aligned} &\frac{\delta\boldsymbol{\theta}_{k+1} - \delta\boldsymbol{\theta}_k}{\delta t} = -\left(\frac{\hat{\boldsymbol{\omega}}_t + \hat{\boldsymbol{\omega}}_{t+1}}{2} - \boldsymbol{b}_\omega\right)^\wedge \delta\boldsymbol{\theta} - \frac{\boldsymbol{n}_{\omega t} + \boldsymbol{n}_{\omega t+1}}{2} - \delta\boldsymbol{b}_\omega \\ &\Rightarrow \delta\boldsymbol{\theta}_{t+1} = \left[\boldsymbol{I} - \left(\frac{\hat{\boldsymbol{\omega}}_t + \hat{\boldsymbol{\omega}}_{t+1}}{2} - \boldsymbol{b}_\omega\right)^\wedge \delta t\right] \delta\boldsymbol{\theta} - \frac{\boldsymbol{n}_{\omega t} + \boldsymbol{n}_{\omega t+1}}{2}\delta t - \delta\boldsymbol{b}_\omega \delta t \end{aligned} \tag{B-50}$$

(2) **速度增量误差** $\delta\boldsymbol{\beta}_t^{b_k}$

同样我们先求中值点的导数值 $\overline{\delta\dot{\boldsymbol{\beta}}}$：

$$\begin{aligned} \delta\dot{\boldsymbol{\beta}}_t &= -\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t - \boldsymbol{b}_a\right)^\wedge \delta\boldsymbol{\theta}_t - \boldsymbol{R}_{b_t}^{b_k}\delta\boldsymbol{b}_a - \boldsymbol{R}_{b_t}^{b_k}\boldsymbol{n}_{at} \\ \delta\dot{\boldsymbol{\beta}}_{t+1} &= -\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^\wedge \delta\boldsymbol{\theta}_{t+1} - \boldsymbol{R}_{b_{t+1}}^{b_k}\delta\boldsymbol{b}_a - \boldsymbol{R}_{b_{t+1}}^{b_k}\boldsymbol{n}_{at+1} \\ \overline{\delta\dot{\boldsymbol{\beta}}} &= \frac{\delta\dot{\boldsymbol{\beta}}_t + \delta\dot{\boldsymbol{\beta}}_{t+1}}{2} \\ &= -\frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t - \boldsymbol{b}_a\right)^\wedge \delta\boldsymbol{\theta}_t - \frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^\wedge \delta\boldsymbol{\theta}_{t+1} - \frac{1}{2}\left(\boldsymbol{R}_{b_t}^{b_k} + \boldsymbol{R}_{b_{t+1}}^{b_k}\right)\delta\boldsymbol{b}_a \\ &\quad - \frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\boldsymbol{n}_{at} - \frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\boldsymbol{n}_{at+1} \end{aligned} \tag{B-51}$$

由于使用了中值积分，这里需要使用 $t+1$ 时刻的角增量误差 $\delta\boldsymbol{\theta}_{t+1}$，将式 B-50带入上式并按照对应的状态量整理得：

$$\begin{aligned}
\delta\dot{\boldsymbol{\beta}}_t = & -\frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t-\boldsymbol{b}_a\right)^{\wedge}\delta\boldsymbol{\theta}_t \\
& -\frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\left\{\left[\boldsymbol{I}-\left(\frac{\hat{\boldsymbol{\omega}}_t+\hat{\boldsymbol{\omega}}_{t+1}}{2}-\boldsymbol{b}_{\omega}\right)^{\wedge}\delta t\right]\delta\boldsymbol{\theta}_t-\frac{\boldsymbol{n}_{\omega t}+\boldsymbol{n}_{\omega t+1}}{2}\delta t-\delta\boldsymbol{b}_{\omega}\delta t\right\} \\
& -\frac{1}{2}\left(\boldsymbol{R}_{b_t}^{b_k}+\boldsymbol{R}_{b_{t+1}}^{b_k}\right)\delta\boldsymbol{b}_a-\frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\boldsymbol{n}_{at}-\frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\boldsymbol{n}_{at+1} \\
= & \left\{-\frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t-\boldsymbol{b}_a\right)^{\wedge}-\frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\left[\boldsymbol{I}-\left(\frac{\hat{\boldsymbol{\omega}}_t+\hat{\boldsymbol{\omega}}_{t+1}}{2}-\boldsymbol{b}_{\omega}\right)^{\wedge}\delta t\right]\right\}\delta\boldsymbol{\theta}_t \\
& -\frac{1}{2}\left(\boldsymbol{R}_{b_t}^{b_k}+\boldsymbol{R}_{b_{t+1}}^{b_k}\right)\delta\boldsymbol{b}_a+\frac{\delta t}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\delta\boldsymbol{b}_{\omega} \\
& -\frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\boldsymbol{n}_{at}-\frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\boldsymbol{n}_{at+1} \\
& +\frac{\delta t}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\boldsymbol{n}_{\omega t}+\frac{\delta t}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\boldsymbol{n}_{\omega t+1}
\end{aligned} \tag{B-52}$$

同样根据导数定义可以得到速度增量误差的递推方程：

$$\begin{aligned}
\delta\boldsymbol{\beta}_{t+1} = & \delta\boldsymbol{\beta}_t+\left\{-\frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t-\boldsymbol{b}_a\right)^{\wedge}-\frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\left[\boldsymbol{I}-\left(\frac{\hat{\boldsymbol{\omega}}_t+\hat{\boldsymbol{\omega}}_{t+1}}{2}-\boldsymbol{b}_{\omega}\right)^{\wedge}\delta t\right]\right\}\delta t\delta\boldsymbol{\theta}_t \\
& -\frac{1}{2}\left(\boldsymbol{R}_{b_t}^{b_k}+\boldsymbol{R}_{b_{t+1}}^{b_k}\right)\delta t\delta\boldsymbol{b}_a+\frac{\delta t^2}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\delta\boldsymbol{b}_{\omega} \\
& -\frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\delta t\boldsymbol{n}_{at}-\frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\delta t\boldsymbol{n}_{at+1} \\
& +\frac{\delta t^2}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\boldsymbol{n}_{\omega t}+\frac{\delta t^2}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\boldsymbol{n}_{\omega t+1}
\end{aligned} \tag{B-53}$$

(3) **位置增量误差** $\delta\boldsymbol{\alpha}_t^{b_k}$ 中值点导数值为 $\overline{\delta\boldsymbol{\alpha}}_t=\frac{\delta\boldsymbol{\beta}_t+\delta\boldsymbol{\beta}_{t+1}}{2}$，进一步可得位置增量误差递推方程为：

$$\delta\boldsymbol{\alpha}_{t+1}=\delta\boldsymbol{\alpha}_t+\frac{\delta\boldsymbol{\beta}_t+\delta\boldsymbol{\beta}_{t+1}}{2}\delta t \tag{B-54}$$

带入速度增量误差表达式有：

$$\begin{aligned}
\delta\boldsymbol{\alpha}_{t+1} = & \delta\boldsymbol{\alpha}_t+\frac{\delta t}{2}\delta\boldsymbol{\beta}_t \\
& +\frac{\delta t}{2}\delta\boldsymbol{\beta}_t+\left\{-\frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t-\boldsymbol{b}_a\right)^{\wedge}-\frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\left[\boldsymbol{I}-\left(\frac{\hat{\boldsymbol{\omega}}_t+\hat{\boldsymbol{\omega}}_{t+1}}{2}-\boldsymbol{b}_{\omega}\right)^{\wedge}\delta t\right]\right\}\frac{\delta t}{2}\delta t\delta\boldsymbol{\theta}_t \\
& -\frac{1}{2}\left(\boldsymbol{R}_{b_t}^{b_k}+\boldsymbol{R}_{b_{t+1}}^{b_k}\right)\delta t\frac{\delta t}{2}\delta\boldsymbol{b}_a+\frac{\delta t^2}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\frac{\delta t}{2}\delta\boldsymbol{b}_{\omega} \\
& -\frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\frac{\delta t}{2}\delta t\boldsymbol{n}_{at}-\frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\frac{\delta t}{2}\delta t\boldsymbol{n}_{at+1} \\
& +\frac{\delta t^2}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\frac{\delta t}{2}\boldsymbol{n}_{\omega t}+\frac{\delta t^2}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1}-\boldsymbol{b}_a\right)^{\wedge}\frac{\delta t}{2}\boldsymbol{n}_{\omega t+1}
\end{aligned} \tag{B-55}$$

整理后得到：

$$
\begin{aligned}
\delta\boldsymbol{\alpha}_{t+1} &= \delta\boldsymbol{\alpha}_t + \delta\boldsymbol{\beta}_t\delta t \\
&+ \left\{ -\frac{1}{4}\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t - \boldsymbol{b}_a\right)^{\wedge}\delta t^2 - \frac{1}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\left[\boldsymbol{I} - \left(\frac{\hat{\boldsymbol{\omega}}_t + \hat{\boldsymbol{\omega}}_{t+1}}{2} - \boldsymbol{b}_{\omega}\right)^{\wedge}\delta t\right]\delta t^2 \right\}\delta\boldsymbol{\theta}_t \\
&- \frac{1}{4}\left(\boldsymbol{R}_{b_t}^{b_k} + \boldsymbol{R}_{b_{t+1}}^{b_k}\right)\delta t^2\delta\boldsymbol{b}_a + \frac{1}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\delta t^3\delta\boldsymbol{b}_{\omega} \\
&- \frac{1}{4}\boldsymbol{R}_{b_t}^{b_k}\delta t^2\boldsymbol{n}_{at} - \frac{1}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\delta t^2\boldsymbol{n}_{at+1} \\
&+ \frac{1}{8}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\delta t^3\boldsymbol{n}_{\omega t} + \frac{1}{8}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\delta t^3\boldsymbol{n}_{\omega t+1}
\end{aligned}
\tag{B-56}
$$

(4) **加速度计零偏误差** $\delta\boldsymbol{b}_a$：根据 IMU 的器件模型有 $\delta\dot{\boldsymbol{b}}_a = \boldsymbol{n}_{\boldsymbol{b}_a}$，递推形式为：$\delta\boldsymbol{b}_{at+1} = \delta\boldsymbol{b}_{at} + \boldsymbol{n}_{\boldsymbol{b}_a}\delta t$。

(5) **陀螺零偏误差** $\delta\boldsymbol{b}_{\omega}$：同样根据 IMU 的器件模型有 $\delta\dot{\boldsymbol{b}}_{\omega} = \boldsymbol{n}_{\boldsymbol{b}_{\omega}}$。

同样，这里我们将上面所有的增量误差递推方程整理为矩阵形式有：

$$
\begin{aligned}
\begin{bmatrix} \delta\boldsymbol{\alpha}_{t+1} \\ \delta\boldsymbol{\theta}_{t+1} \\ \delta\boldsymbol{\beta}_{t+1} \\ \delta\boldsymbol{b}_{at+1} \\ \delta\boldsymbol{b}_{\omega t+1} \end{bmatrix} &= \begin{bmatrix} \boldsymbol{I} & f_{01} & \delta t\boldsymbol{I} & f_{03} & f_{04} \\ 0 & f_{11} & 0 & 0 & -\delta t\boldsymbol{I} \\ 0 & f_{21} & \boldsymbol{I} & f_{23} & f_{24} \\ 0 & 0 & 0 & \boldsymbol{I} & 0 \\ 0 & 0 & 0 & 0 & \boldsymbol{I} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\alpha}_t \\ \delta\boldsymbol{\theta}_t \\ \delta\boldsymbol{\beta}_t \\ \delta\boldsymbol{b}_{at} \\ \delta\boldsymbol{b}_{\omega t} \end{bmatrix} \\
&+ \begin{bmatrix} v_{00} & v_{01} & v_{02} & v_{03} & 0 & 0 \\ 0 & -\frac{\delta t}{2}\boldsymbol{I} & 0 & -\frac{\delta t}{2}\boldsymbol{I} & 0 & 0 \\ -\frac{\boldsymbol{R}_{b_t}^{b_k}\delta t}{2} & v_{21} & -\frac{\boldsymbol{R}_{b_{t+1}}^{b_k}\delta t}{2} & v_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & \boldsymbol{I}\delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & \boldsymbol{I}\delta t \end{bmatrix} \begin{bmatrix} \boldsymbol{n}_{at} \\ \boldsymbol{n}_{\omega t} \\ \boldsymbol{n}_{at+1} \\ \boldsymbol{n}_{\omega t+1} \\ \boldsymbol{n}_{\boldsymbol{b}_a} \\ \boldsymbol{n}_{\boldsymbol{b}_{\omega}} \end{bmatrix}
\end{aligned}
\tag{B-57}
$$

$\boldsymbol{F}$ 矩阵中对应元素具体表达式如下：

$$
\begin{aligned}
f_{01} &= \left\{ -\frac{1}{4}\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t - \boldsymbol{b}_a\right)^{\wedge}\delta t^2 - \frac{1}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\left[\boldsymbol{I} - \left(\frac{\hat{\boldsymbol{\omega}}_t + \hat{\boldsymbol{\omega}}_{t+1}}{2} - \boldsymbol{b}_{\omega}\right)^{\wedge}\delta t\right]\delta t^2 \right\} \\
f_{03} &= -\frac{1}{4}\left(\boldsymbol{R}_{b_t}^{b_k} + \boldsymbol{R}_{b_{t+1}}^{b_k}\right)\delta t^2 \\
f_{04} &= \frac{1}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\delta t^3 \\
f_{11} &= \left[\boldsymbol{I} - \left(\frac{\hat{\boldsymbol{\omega}}_t + \hat{\boldsymbol{\omega}}_{t+1}}{2} - \boldsymbol{b}_{\omega}\right)^{\wedge}\delta t\right] \\
f_{21} &= \left\{ -\frac{1}{2}\boldsymbol{R}_{b_t}^{b_k}\left(\hat{\boldsymbol{a}}_t - \boldsymbol{b}_a\right)^{\wedge} - \frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\left[\boldsymbol{I} - \left(\frac{\hat{\boldsymbol{\omega}}_t + \hat{\boldsymbol{\omega}}_{t+1}}{2} - \boldsymbol{b}_{\omega}\right)^{\wedge}\delta t\right] \right\}\delta t \\
f_{23} &= -\frac{1}{2}\left(\boldsymbol{R}_{b_t}^{b_k} + \boldsymbol{R}_{b_{t+1}}^{b_k}\right)\delta t \\
f_{24} &= \frac{1}{2}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\delta t^2
\end{aligned}
\tag{B-58}
$$

$\boldsymbol{V}$ 矩阵中对应元素具体表达式如下：

$$
\begin{aligned}
v_{00} &= -\frac{1}{4}\boldsymbol{R}_{b_t}^{b_k}\delta t^2 \\
v_{01} &= \frac{1}{8}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\delta t^3 \\
v_{02} &= -\frac{1}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\delta t^2 \\
v_{03} &= \frac{1}{8}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\delta t^3 \\
v_{21} &= \frac{1}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\delta t^2 \\
v_{23} &= \frac{1}{4}\boldsymbol{R}_{b_{t+1}}^{b_k}\left(\hat{\boldsymbol{a}}_{t+1} - \boldsymbol{b}_a\right)^{\wedge}\delta t^2
\end{aligned}
\tag{B-59}
$$

C C++ 多线程编程

C.1 开辟线程

多线程编程主要为了解决程序中的并发问题，计算机术语中的" 并发"，指的是在单个系统里同时执行多个独立的活动，而不是顺序的一个接一个的执行。对于单核 CPU 来说，在某个时刻只可能处理一个任务，但它却不是完全执行完一个任务再执行一个下一任务，而是一直在任务间切换，每个任务完成一点就去执行下一个任务，看起来就像任务在并行发生，虽然不是严格的同时执行多个任务，但是我们仍然称之为并发 (concurrency)。真正的并发是在在多核 CPU 上，能够真正的同时执行多个任务，称为硬件并发 (hardware concurrency)。

C++ 处理并发任务的方式包括进程并发和线程并发两种；进程并发更为安全，可部署在分布式系统上，但通信机制较为复杂，需要耗费更多的系统资源；线程并发基于共享内存实现，由于

线程间的大量数据可以共享，线程并发具有更小的资源消耗和运行速度，但同时也需要更为范围的程序语言维护以防出错。图 C-2给出了进程并发于线程并发的区别。C++98 标准中并没有线程

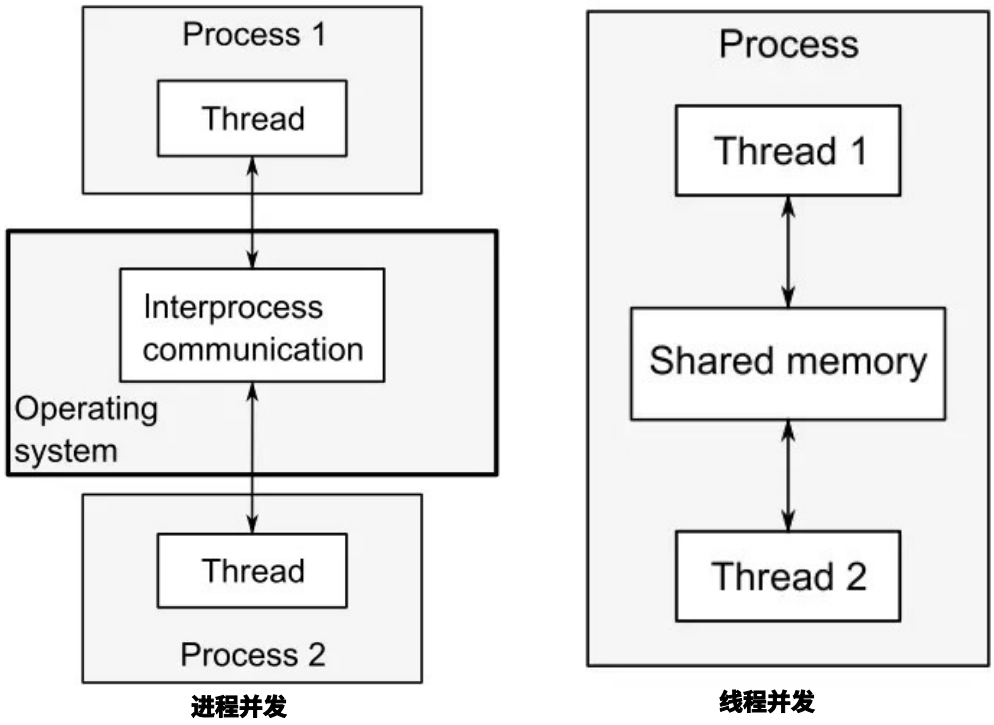

图 C-2: 进程并发与线程并发

库的存在，而在 C++11 中终于提供了多线程的标准库，提供了管理线程、保护共享数据、线程间同步操作、原子操作等类。多线程库对应的头文件是 <**thread**>，类名为 **std::thread**。现在假设我们有一个函数 function1()，单线程串行运行该程序的代码为：

```
1#include <iostream>
2#include <thread>
3
4void function_1() {
5   std::cout << "I'm␣function_1()" << std::endl;
6}
7
8int main() {
9   function_1();
```

```
10    return 0;
11 }
```

这里我们并没有使用 thread 的相关功能去开辟和管理线程，这是因为 main() 函数本身即为一个线程，成为主线程，所有的程序会在主线程中顺序运行。若我们向让 function1() 在其他线程中运行，可以按如下方式改写：

```
1 #include <iostream>
2 #include <thread>
3
4 void function_1() {
5    std::cout << "I'm function_1()" << std::endl;
6 }
7
8 int main() {
9    std::thread t1(function_1);//开辟一个线程t1并运行函数
       function1
10   // do other things
11   std::cout << "main() before" << std::endl;
12   std::cout << "main() before" << std::endl;
13   std::cout << "main() before:" << std::endl;
14   t1.join();//
15   std::cout << "main() after" << std::endl;
16   return 0;
17 }
```

代码分析：

(1) 首先使用 std::thread name(fcn) 函数创建一个**线程对象**，fcn 称为线程对象的**入口函数**，该函数的运行周期即为线程的生命周期。

(2) 当线程对象 t1 创建后，对应的线程函数 function1() 即开始同步运行，函数运行结束即标志着**线程**结束，但**线程对象**并未释放；

(3) main 线程和 t1 线程是同步执行的，但两个线程并不是同时结束的，因此需要在 main 函数中设定线程的执行类型。线程的类型包括 join() 和 detach() 两种；join() 称为线程等待函数，

main 线程在同步运行至 join() 函数时，会判断对应的 t1 线程是否运行完成；若完成子继续运行 main 线程中 join() 后的内容，否则等待 t1 线程运行。与之相对，detach() 称为线程分离函数，main 线程运行至 detach() 时会将 t1 线程分离出去，随即继续运行 main 线程的后续内容。

C.2 线程构造函数

线程构造函数 std::thread() 是一个可变参数函数，第一个参数为线程的入口函数，后面参数以此为入口函数的对应参数，例如：

```
1 // 普通函数 无参
2 void function_1() {
3 }
4
5 // 普通函数 1个参数
6 void function_2(int i) {
7 }
8
9 // 普通函数 2个参数
10 void function_3(int i, std::string m) {
11 }
12
13 std::thread t1(function_1);
14 std::thread t2(function_2, 1);
15 std::thread t3(function_3, 1, "hello");
16
17 t1.join();
18 t2.join();
19 t3.join();
```

需要注意的是，线程的入口函数不接受重载函数，重载函数会使得编译器无法决定正确的函数类型从而引发编译错误。

此外，入口函数还可以是仿函数、匿名函数和 std::function 对象。

(1) 仿函数：

```
1     // 仿函数
2     class Fctor {
3     public:
4     // 具有一个参数
5     void operator() () {
6
7     }
8     };
9     Fctor f;
10    std::thread t1(f);
11    // std::thread t2(Fctor()); // 编译错误
12    std::thread t3((Fctor())); // ok
13    std::thread t4{Fctor()}; // ok
```

(2) 匿名函数:

```
1     std::thread t1([](){
2     std::cout << "hello" << std::endl;
3     });
4
5     std::thread t2([](std::string m){
6     std::cout << "hello␣" << m << std::endl;
7     }, "world");
```

(3) std::function:

```
1     class A{
2     public:
3     void func1(){
4     }
5
6     void func2(int i){
7     }
```

```
8     void func3(int i, int j){
9     }
10    };
11
12    A a;
13    std::function<void(void)> f1 = std::bind(&A::func1, &a);
14    std::function<void(void)> f2 = std::bind(&A::func2, &a, 1);
15    std::function<void(int)> f3 = std::bind(&A::func2, &a, std
          ::placeholders::_1);
16    std::function<void(int)> f4 = std::bind(&A::func3, &a, 1,
          std::placeholders::_1);
17    std::function<void(int, int)> f5 = std::bind(&A::func3, &a,
          std::placeholders::_1, std::placeholders::_2);
18
19    std::thread t1(f1);
20    std::thread t2(f2);
21    std::thread t3(f3, 1);
22    std::thread t4(f4, 1);
23    std::thread t5(f5, 1, 2);
```

注 1：需要注意的是，线程入口函数传入的参数只有**传值**一种类型，即便入口函数的参数是引用形式，为了保证线程数据的安全性，新开辟线程也会在构造线程兑现对传入的数值进行拷贝。
注 2：线程对象之间是不能复制的，只能移动，移动的意思是，将线程的所有权在 std::thread 实例间进行转移。

```
1  void some_function();
2  void some_other_function();
3  std::thread t1(some_function);
4  // std::thread t2 = t1; // 编译错误
5  std::thread t2 = std::move(t1); //只能移动 t1内部已经没有线程
       了
6  t1 = std::thread(some_other_function); // 临时对象赋值 默认就
       是移动操作
7  std::thread t3;
```

```
8   t3 = std::move(t2); // t2内部已经没有线程了
9   t1 = std::move(t3); // 程序将会终止，因为t1内部已经有一个线程
        在管理了
```

C.3 线程与竞争条件

并发代码中最常见的错误之一就是竞争条件 (race condition)。而其中最常见的就是数据竞争 (data race)，从整体上来看，所有线程之间共享数据的问题，都是修改数据导致的，如果所有的共享数据都是只读的，就不会发生问题。但是这是不可能的，大部分共享数据都是要被修改的。

std::cout() 就是一个典型的共享资源对象，其操作基于数据流实现：

```
1#include <iostream>
2#include <thread>
3#include <string>
4using namespace std;
5
6// 普通函数 无参
7void function_1() {
8for(int i=0; i>-100; i--)
9   cout << "From␣t1:␣" << i << endl;
10}
11
12int main()
13{
14   std::thread t1(function_1);
15
16   for(int i=0; i<100; i++)
17      cout << "From␣main:␣" << i << endl;
18
19   t1.join();
20   return 0;
21}
```

在上面这里 demo 中，运行结果会出现如下的混乱情况，这是因为 cout 的数据流在两个线程中是共享的，两个线程执行 cout 取出数据的时机是完全随机的：

```
From main: 0
From main: 1From t1:
From main: 2
```

c++ 中提供了解决这一问题的方式——互斥锁 (Mutex)，其包含在头文件 <**mutex**> 中，在使用共享资源前使用 std::mutex.lock() 锁定线程，使用完毕后使用 std::mutex.unlock() 解锁线程，即可保护共享资源对象：

```
#include <iostream>
#include <thread>
#include <string>
#include <mutex>
using namespace std;

std::mutex mu;
// 使用锁保护
void shared_print(string msg, int id) {
  mu.lock(); // 上锁
  cout << msg << id << endl;
  mu.unlock(); // 解锁
}

void function_1() {
  for(int i=0; i>-100; i--)
    shared_print(string("From t1: "), i);
}

```

```
20 int main()
21 {
22   std::thread t1(function_1);
23
24   for(int i=0; i<100; i++)
25     shared_print(string("From␣main:␣"), i);
26
27   t1.join();
28   return 0;
29 }
```

但由用户手动加锁与解锁存在一个隐藏问题，即当 lock 于 unlock 之间的代码出现异常退出时，系统不会自动将共享资源解锁，时候后续需要使用该共享资源的线程秩序堵塞。C++ 标准库提供了 lock_guard 模板类解决这一问题，该模板类是 c++ 中常见的 RAII 技术，即获取资源即初始化 (Resource Acquisition Is Initialization) 技术。该模板类在创建类对象的时候会自动对线程加锁，在对象析构时，则会自动对线程解锁。

```
1 void shared_print(string msg, int id) {
2   //构造的时候帮忙上锁
3   std::lock_guard<std::mutex> guard(mu);
4
5   cout << msg << id << endl;
6   //即便代码出现异常退出，由于函数运行结束，函数内创建的
      lock_guard对象
7   //释放时会自动调用析构函数，保证线程解锁
8 }
```

同时，由于 mutex 本身是一个全局变量，但其服务对象仅为 shared_print 函数，因此可以将两者分别封装为类的成员变量和成员函数，以优化代码：

```
1 #include <iostream>
```

```
2#include <thread>
3#include <string>
4#include <mutex>
5#include <fstream>
6using namespace std;
7
8std::mutex mu;
9class LogFile {
10std::mutex m_mutex;
11ofstream f;
12public:
13LogFile() {
14f.open("log.txt");
15}
16~LogFile() {
17f.close();
18}
19void shared_print(string msg, int id) {
20std::lock_guard<std::mutex> guard(mu);
21f << msg << id << endl;
22}
23};
24
25void function_1(LogFile& log) {
26for(int i=0; i>-100; i--)
27log.shared_print(string("From␣t1:␣"), i);
28}
29
30int main()
31{
32LogFile log;
33//log对象不能拷贝，需要使用ref传入原引用
34std::thread t1(function_1, std::ref(log));
35
```

```
36 for(int i=0; i<100; i++)
37 log.shared_print(string("From main: "), i);
38
39 t1.join();
40 return 0;
41 }
```

上一节我们提到过，线程函数的参数本质是拷贝而非赋值。这里由于 ofstream 对象不能直接拷贝，需要使用 std::ref() 函数将原对象的引用传入。

C.4 死锁

将 mutex 上锁而不解锁的情况称为**死锁**，这种情况下使用 lock_guard 可以保证析构的时候能够释放锁，然而，当一个操作需要使用两个互斥锁的时候，仅仅使用 lock_guard 并不能保证不会发生死锁，如下面的例子：

```
1#include <iostream>
2#include <thread>
3#include <string>
4#include <mutex>
5#include <fstream>
6using namespace std;
7
8class LogFile {
9   std::mutex _mu;
10  std::mutex _mu2;
11  ofstream f;
12public:
13  LogFile() {
14    f.open("log.txt");
15  }
16  ~LogFile() {
17    f.close();
18  }
```

```
19   void shared_print(string msg, int id) {
20     std::lock_guard<std::mutex> guard(_mu);
21     std::lock_guard<std::mutex> guard2(_mu2);
22     f << msg << id << endl;
23     cout << msg << id << endl;
24   }
25   void shared_print2(string msg, int id) {
26     std::lock_guard<std::mutex> guard(_mu2);
27     std::lock_guard<std::mutex> guard2(_mu);
28     f << msg << id << endl;
29     cout << msg << id << endl;
30   }
31 };
32
33 void function_1(LogFile& log) {
34   for(int i=0; i>-100; i--)
35     log.shared_print2(string("From␣t1:␣"), i);
36 }
37
38 int main()
39 {
40   LogFile log;
41   std::thread t1(function_1, std::ref(log));
42
43   for(int i=0; i<100; i++)
44     log.shared_print(string("From␣main:␣"), i);
45
46   t1.join();
47   return 0;
48 }
```

该 demo 中由于同时使用了两个互斥锁，在某些状况下会出现嵌套死锁，应尽量避免。

C.5 unique_lock

使用互斥锁虽然保证了共享数据的唯一性，但亦将并行操作变为了串行操作，降低了代码效率，因此需要谨慎的设计上锁的代码块。如果一段代码中有多个代码块需要频繁的加锁与解锁，由于 lock_guard 只会在创建时上锁，在析构时解锁，就需要创建多个局部 lock_guard 对象，例如：

```
std::mutex _mu;
{
  std::lock_guard<std::mutex> guard(_mu);
  \\code....
}
{
  std::lock_guard<std::mutex> guard(_mu);
  \\code....
}
{
  std::lock_guard<std::mutex> guard(_mu);
  \\code....
}
```

这无疑增加了代码的复杂度，降低了可阅读性。而使用 unique_lock 则可有效改善这一问题。unique_lock 提供了 lock() 和 unlock() 接口，能记录现在处于上锁还是没上锁状态，在析构的时候，会根据当前状态来决定是否要进行解锁（lock_guard 就一定会解锁）。上面的代码修改如下：

```
class LogFile {
  std::mutex _mu;
  ofstream f;
  public:
    LogFile() {
      f.open("log.txt");
    }
    ~LogFile() {
      f.close();
```

```
10      }
11      void shared_print(string msg, int id) {
12
13          std::unique_lock<std::mutex> guard(_mu);
14          //do something 1
15          guard.unlock(); //临时解锁
16          //do something 2
17          guard.lock(); //继续上锁
18          // do something 3
19          f << msg << id << endl;
20          cout << msg << id << endl;
21          // 结束时析构guard会临时解锁
22          // 这句话可要可不要，不写，析构的时候也会自动执行
23          // guard.ulock();
24      }
25  };
```

可以看到，unique_lock 记录了当前锁的状态，在无需加锁的代码区域，可以使用 unlock 借口临时解锁，随后再重新加锁。但由于需要一直维护锁的状态，unique_lock 的执行效率比 lock_guard 要低，因此能使用 lock_guard 解决的代码应尽量使用 lock_guard。

同时需要注意的是，lock_guard 和 unique_lock 均不能复制，但 unique_lock 可以移动。

C.6 条件变量

在 C++11 以后的标准中，条件变量是实现线程同步管理的另一个有效方式，其通常和 unique_lock 一同配套使用以避免线程间的竞争。使用条件变量需要包含头文件 **condition_variable**，其主要包括两个重要的成员函数机制，等待和唤醒：

(1) 等待
(1) wait()：阻塞当前线程直到条件变量唤醒
(2) wait_for()：阻塞当前线程直到条件变量唤醒或到达一定时间
(3) wait_until()：阻塞当前线程直到条件变量唤醒或到达指定时间点
(2) 唤醒
(1) notify_one()：随机唤醒一个等待线程

(2) notify_all()：唤醒所有等待线程

其中，所有 wait 函数接受参数均为 std::unique_lock<std::mutex> 对象，且可以选择使用微词 pred 避免虚假唤醒，即：

```
1 void wait( std::unique_lock<std::mutex>& lock );
2 //Predicate 谓词函数，可以普通函数或者lambda表达式
3 template< class Predicate >
4 void wait( std::unique_lock<std::mutex>& lock, Predicate pred )
    ;
```

参考文献