GNSS/INS组合导航开源代码

KF-GINS分享交流

王立强、陈起金

武汉大学多源智能导航实验室（i2Nav）

www.i2nav.com

https://github.com/i2Nav-WHU

2023年5月14日

i2Nav微信公众号

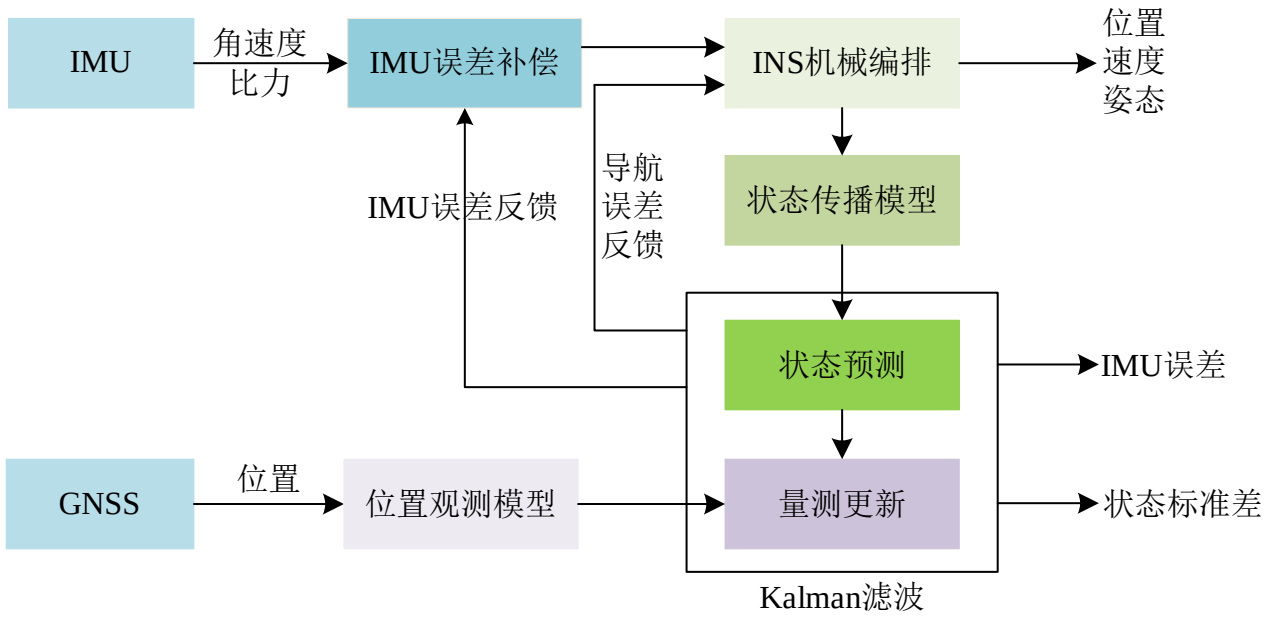
i2Nav开源算法交流QQ群

目 录

KF-GINS概述

KF-GINS在卡尔曼滤波框架下实现了GNSS、INS松组合算法

- 开源代码、算法讲义、课程视频
- 扩展(误差状态)卡尔曼滤波架构，包括IMU误差补偿、惯性导航解算、Kalman滤波、误差反馈等
- 采用**21维系统误差状态**，包括位置误差、速度误差、姿态误差、IMU零偏、比例因子
- 姿态误差采用**Phi角模型**，速度、位置误差定义在导航坐标系下
- 惯性导航解算采用基于角速度和加速度线性变化假设的**双子样机械编排算法**，补偿了姿态圆锥效应、速度的旋转效应和划桨效应



KF-GINS系统框架

- 牛小骥, 陈起金, “惯性导航原理与GNSS/INS组合导航课程讲义”，武汉大学多源智能导航实验室网站
- 牛小骥, 陈起金, “惯性导航原理与GNSS/INS组合导航课程视频”，bilibili网站“i2Nav”

目 录

GNSS/INS松组合算法

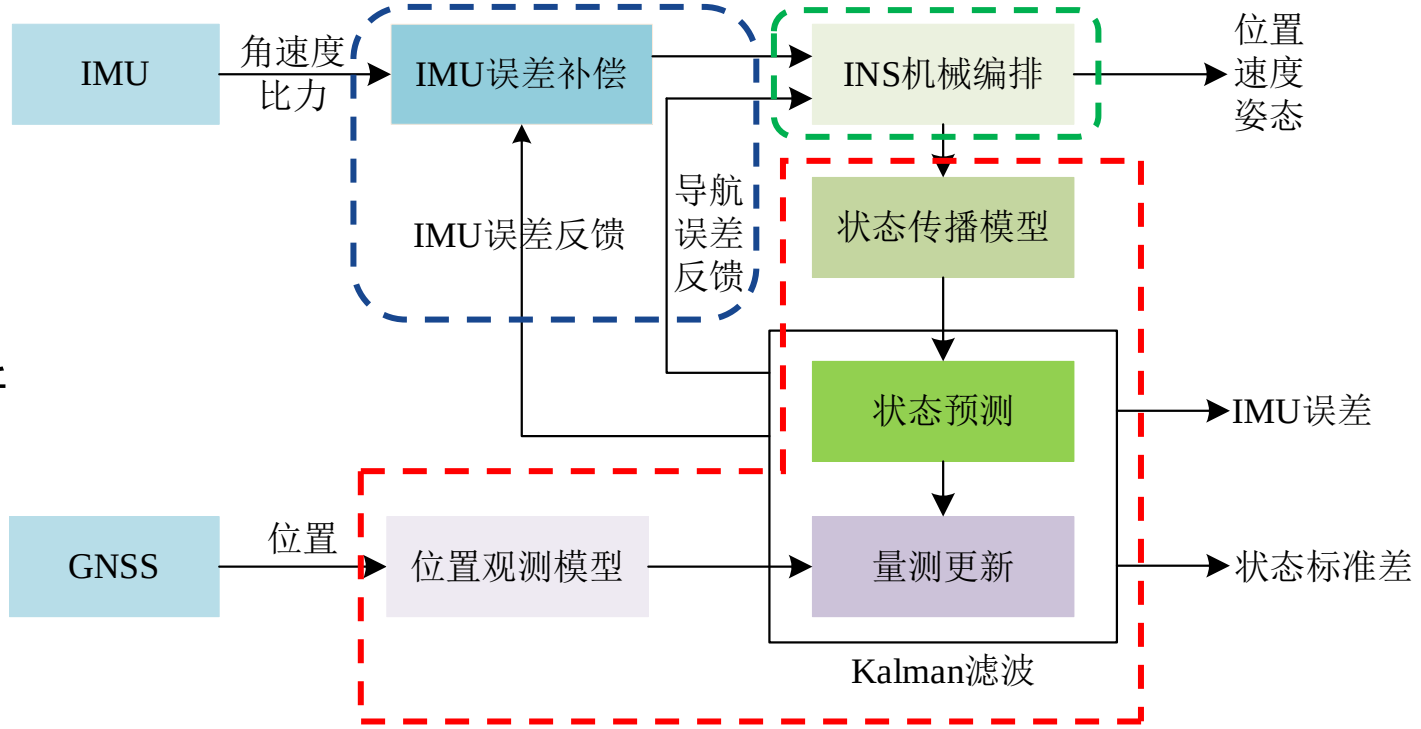
惯性导航算法

- INS机械编排

组合导航算法

- 卡尔曼滤波：状态预测，量测更新
- INS状态传播模型
- GNSS位置观测模型

误差补偿、反馈算法



KF-GINS系统框架

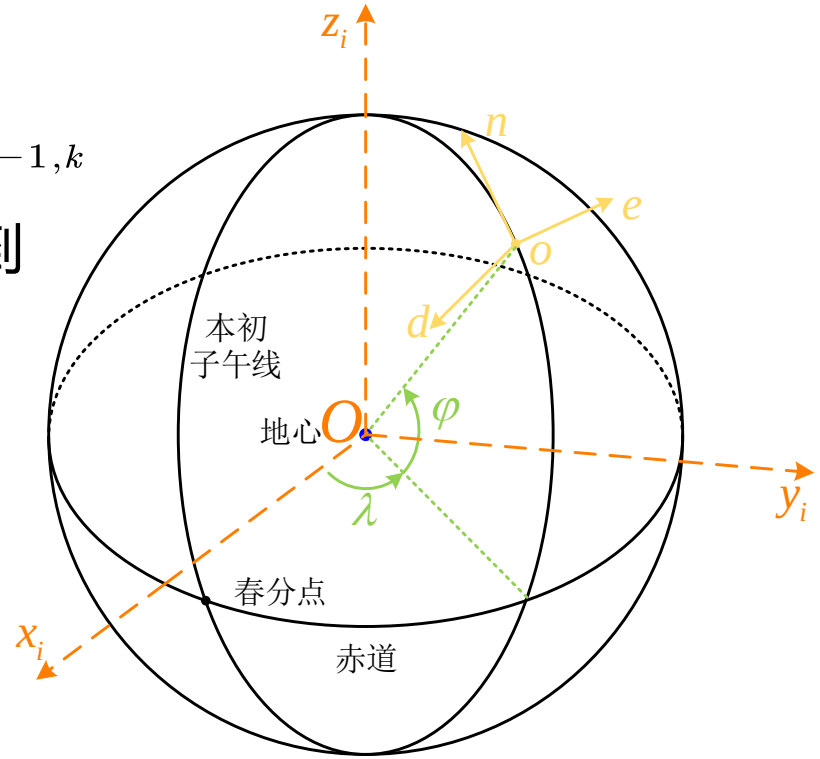
捷联惯性导航算法

IMU测量值

- 速率型数据：加速度和角速度：$\boldsymbol{f}_{ib}^{b}$、 $\boldsymbol{\omega}_{ib}^{b}$
- **增量型数据**：采样间隔内速度增量和角度增量：$\Delta\boldsymbol{v}_{k-1,k}^{b}$、 $\Delta\boldsymbol{\theta}_{k-1,k}^{b}$
- 高精度惯导输出增量数据；通过模拟积分，或高频采样积分得到

坐标系

- 载体坐标系(b系)，IMU几何中心为原点，前向、右向和垂向（下）
- 导航坐标系(n系)，导航结果描述对象为原点，北向、东向和地向
- 大地坐标系，纬度、经度、高程表示绝对位置
- (地心)惯性坐标系(i系)，惯性传感器测量值的参考坐标系



常用参考坐标系

新手入门系列4——MEMS IMU**原始数据采集和时间同步**的那些坑
http://i2nav.com/index/newListDetail_zw.do?newskind_id=13a8654e060c40c69e5f3d4c13069078&newsinfo_id=2e05f5cdac6b4725b8bfe54a689c7add

捷联惯性导航算法

简化IMU微分方程

$$\dot{\boldsymbol{p}}_{wb}^{w}=\boldsymbol{v}_{wb}^{w}$$

$$\dot{\boldsymbol{v}}_{wb}^{w}=\boldsymbol{C}_{b}^{w}\boldsymbol{f}_{ib}^{b}+\boldsymbol{g}_{l}^{w}$$

$$\dot{\boldsymbol{C}}_{b}^{w}=\boldsymbol{C}_{b}^{w}(\boldsymbol{\omega}_{wb}^{b}\times)$$

$$\dot{\varphi}=\frac{v_N}{R_M+h},\ \dot{\lambda}=\frac{v_E}{(R_N+h)\cos\varphi},\ \dot{h}=-v_D$$

$$\dot{\boldsymbol{v}}_{eb}^{n}=\boldsymbol{C}_{b}^{n}\boldsymbol{f}_{ib}^{b}+\boldsymbol{g}_{l}^{n}-(2\boldsymbol{\omega}_{ie}^{n}+\boldsymbol{\omega}_{en}^{n})\times\boldsymbol{v}_{eb}^{n}$$

$$\dot{\boldsymbol{C}}_{b}^{n}=\boldsymbol{C}_{b}^{n}[\boldsymbol{\omega}_{nb}^{b}]_{\times}=\boldsymbol{C}_{b}^{n}[\boldsymbol{\omega}_{ib}^{b}]_{\times}-[\boldsymbol{\omega}_{in}^{n}]_{\times}\boldsymbol{C}_{b}^{n}$$

捷联惯性导航微分方程

简化中值积分

$$\boldsymbol{v}_{wb,k}^{w}=\boldsymbol{v}_{wb,k-1}^{w}+\boldsymbol{C}_{b,k-1}^{w}\Delta\boldsymbol{v}_k+\boldsymbol{g}_{l}^{w}\Delta t_k$$

$$\boldsymbol{p}_{wb,k}^{w}=\boldsymbol{p}_{wb,k-1}^{w}+\frac{1}{2}(\boldsymbol{v}_{wb,k-1}^{w}+\boldsymbol{v}_{wb,k}^{w})\Delta t_k$$

$$\boldsymbol{q}_{b_k}^{w}=\boldsymbol{q}_{b_{k-1}}^{w}\otimes\boldsymbol{q}_{b_k}^{b_{k-1}}=\boldsymbol{q}_{b_{k-1}}^{w}\otimes\begin{bmatrix}1\\ \Delta\theta_k/2\end{bmatrix}$$

$$\boldsymbol{v}_{eb,k}^{n}=\boldsymbol{v}_{eb,k-1}^{n}+\underbrace{\left[\boldsymbol{I}-\frac{1}{2}(\boldsymbol{\zeta}_{n(k-1)n(k)}\times)\right]\boldsymbol{C}_{b(k-1)}^{n(k-1)}\Delta\boldsymbol{v}_{f,k}^{b(k-1)}}_{\text{比力积分项}}+\underbrace{\boldsymbol{g}_{l}^{n}\Delta t_k-(2\boldsymbol{\omega}_{ie}^{n}+\boldsymbol{\omega}_{en}^{n})\times\boldsymbol{v}_{eb}^{n}\big|_{t=t_{k-1/2}}\Delta t_k}_{\text{重力/哥氏积分项}}$$

$$\boldsymbol{\zeta}_{n(k-1)n(k)}\approx\boldsymbol{\omega}_{in}^{n}\big|_{t=t_{k-1/2}}\Delta t_k$$

$$\Delta\boldsymbol{v}_{f,k}^{b(k-1)}=\Delta\boldsymbol{v}_k+\frac{1}{2}\Delta\boldsymbol{\theta}_k\times\Delta\boldsymbol{v}_k+\frac{1}{12}(\Delta\boldsymbol{\theta}_{k-1}\times\Delta\boldsymbol{v}_k+\Delta\boldsymbol{v}_{k-1}\times\Delta\boldsymbol{\theta}_k)$$

捷联惯导速度更新

哥氏积分项划桨、旋转效应

捷联惯导位置更新

$$h_k=h_{k-1}-\frac{1}{2}(v_{D,k-1}+v_{D,k})\Delta t_k$$

$$\varphi_k=\varphi_{k-1}+\frac{v_{N,k-1}+v_{N,k}}{2(R_{M,k-\frac{1}{2}}+h_{k-\frac{1}{2}})}\Delta t_k$$

$$\lambda_k=\lambda_{k-1}+\frac{v_{E,k-1}+v_{E,k}}{2(R_{N,k-\frac{1}{2}}+h_{k-\frac{1}{2}})\cos\varphi_{k-\frac{1}{2}}}\Delta t_k$$

$$\boldsymbol{q}_{b_k}^{n_k}=\boldsymbol{q}_{n_{k-1}}^{n_k}\boldsymbol{q}_{b_{k-1}}^{n_{k-1}}\boldsymbol{q}_{b_k}^{b_{k-1}}$$

$$\boldsymbol{q}_{n_{k-1}}^{n_k}\leftarrow\boldsymbol{\zeta}_k\approx\boldsymbol{\omega}_{in}^{n}\big|_{t=t_{k-1/2}}\Delta t_k$$

$$\boldsymbol{q}_{b_k}^{b_{k-1}}\leftarrow\boldsymbol{\phi}_k=\Delta\boldsymbol{\theta}_k+\frac{1}{12}\Delta\boldsymbol{\theta}_{k-1}\times\Delta\boldsymbol{\theta}_k$$

等效旋转矢量更新圆锥效应

捷联惯导姿态更新

GNSS/INS组合导航

状态方程

- 扩展卡尔曼滤波(误差状态卡尔曼滤波)
- 扰动形式求解误差微分方程，**n系下进行位置误差扰动**
- **建模IMU零偏、比例因子误差**为一阶高斯马尔可夫过程
- 考虑**IMU测量白噪声，零偏和比例因子模型的驱动白噪声**

系统变量和过程噪声

$$\boldsymbol{x}=[(\delta\boldsymbol{r}^n)^T\quad(\delta\boldsymbol{v}^n)^T\quad\boldsymbol{\phi}^T\quad\boldsymbol{b}_g^T\quad\boldsymbol{b}_a^T\quad\boldsymbol{s}_g^T\quad\boldsymbol{s}_a^T]$$

$$\boldsymbol{w}_t=\mathrm{diag}[\boldsymbol{w}_g^T\quad\boldsymbol{w}_a^T\quad\boldsymbol{w}_{gb}^T\quad\boldsymbol{w}_{ab}^T\quad\boldsymbol{w}_{gs}^T\quad\boldsymbol{w}_{as}^T]$$

导航状态扰动

$$\hat{\boldsymbol{r}}^n=\boldsymbol{r}^n+\delta\boldsymbol{r}^n$$

$$\hat{\boldsymbol{v}}^n=\boldsymbol{v}^n+\delta\boldsymbol{v}^n$$

$$\hat{\boldsymbol{C}}_b^n=[\boldsymbol{I}-(\boldsymbol{\phi}\times)]\boldsymbol{C}_b^n$$

微分方程

$$\delta\dot{\boldsymbol{r}}^n=-\boldsymbol{\omega}_{en}^n\times\delta\boldsymbol{r}^n+\delta\boldsymbol{\theta}\times\boldsymbol{v}^n+\delta\boldsymbol{v}^n$$

$$\delta\dot{\boldsymbol{v}}^n=\boldsymbol{f}^n\times\boldsymbol{\phi}-(2\boldsymbol{\omega}_{ie}^n+\boldsymbol{\omega}_{en}^n)\times\delta\boldsymbol{v}^n+\boldsymbol{v}^n\times(2\delta\boldsymbol{\omega}_{ie}^n+\delta\boldsymbol{\omega}_{en}^n)+\delta\boldsymbol{g}_l^n+\boldsymbol{C}_b^n\delta\boldsymbol{f}^b$$

$$\dot{\boldsymbol{\phi}}=-\boldsymbol{\omega}_{in}^n\times\boldsymbol{\phi}+\delta\boldsymbol{\omega}_{in}^n-\boldsymbol{C}_b^n\delta\boldsymbol{\omega}_{ib}^b$$

IMU误差

$$\delta\boldsymbol{\omega}_{ib}^b=\boldsymbol{b}_g+diag(\boldsymbol{s}_g)\boldsymbol{\omega}_{ib}^b+\boldsymbol{w}_g$$

$$\delta\boldsymbol{f}_{ib}^b=\boldsymbol{b}_a+diag(\boldsymbol{s}_a)\boldsymbol{f}_{ib}^b+\boldsymbol{w}_a$$

IMU误差微分方程

$$\dot{\boldsymbol{b}}_g=-\boldsymbol{b}_g/T_{gb}+\boldsymbol{\omega}_{gb}$$

$$\dot{\boldsymbol{s}}_g=-\boldsymbol{s}_g/T_{ga}+\boldsymbol{\omega}_{gs}$$

$$\dot{\boldsymbol{b}}_a=-\boldsymbol{b}_a/T_{ab}+\boldsymbol{\omega}_{ab}$$

$$\dot{\boldsymbol{s}}_a=-\boldsymbol{s}_a/T_{as}+\boldsymbol{\omega}_{as}$$

GNSS/INS组合导航

误差微分方程

$$\delta\dot{\boldsymbol{r}}^n = \boldsymbol{F}_{rr}\delta\boldsymbol{r}^n + \delta\boldsymbol{v}^n$$

$$\delta\dot{\boldsymbol{v}}^n = \boldsymbol{F}_{vr}\delta\boldsymbol{r}^n + \boldsymbol{F}_{vv}\delta\boldsymbol{v}^n + \boldsymbol{f}^n \times \boldsymbol{\phi} + \boldsymbol{C}_b^n \boldsymbol{b}_a + \boldsymbol{C}_b^n diag(\boldsymbol{f}_{ib}^b)\boldsymbol{s}_a + \boldsymbol{C}_b^n \boldsymbol{w}_a$$

$$\dot{\boldsymbol{\phi}} = \boldsymbol{F}_{\phi r}\delta\boldsymbol{r}^n + \boldsymbol{F}_{\phi v}\delta\boldsymbol{v}^n - \boldsymbol{\omega}_{in}^n \times \boldsymbol{\phi} - \boldsymbol{C}_b^n \boldsymbol{b}_g + \boldsymbol{C}_b^n diag(\boldsymbol{\omega}_{ib}^b)\boldsymbol{s}_g + \boldsymbol{C}_b^n \boldsymbol{w}_g$$

$$\dot{\boldsymbol{b}}_g = -\boldsymbol{b}_g/T_{gb} + \boldsymbol{\omega}_{gb}$$

$$\dot{\boldsymbol{s}}_g = -\boldsymbol{s}_g/T_{ga} + \boldsymbol{\omega}_{gs}$$

$$\dot{\boldsymbol{b}}_a = -\boldsymbol{b}_a/T_{ab} + \boldsymbol{\omega}_{ab}$$

$$\dot{\boldsymbol{s}}_a = -\boldsymbol{s}_a/T_{as} + \boldsymbol{\omega}_{as}$$

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{F}_{rr} & \boldsymbol{I}_{3\times3} & 0 & 0 & 0 & 0 & 0 \\ \boldsymbol{F}_{vr} & \boldsymbol{F}_{vv} & [(\boldsymbol{C}_b^n \boldsymbol{f}^b)\times] & 0 & \boldsymbol{C}_b^n & 0 & \boldsymbol{C}_b^n \mathrm{diag}(\boldsymbol{f}^b) \\ \boldsymbol{F}_{\phi r} & \boldsymbol{F}_{\phi v} & -(\boldsymbol{\omega}_{in}^n \times) & -\boldsymbol{C}_b^n & 0 & -\boldsymbol{C}_b^n \mathrm{diag}(\boldsymbol{\omega}_{ib}^b) & 0 \\ 0 & 0 & 0 & -1/T_{gb}\boldsymbol{I}_{3\times3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1/T_{ab}\boldsymbol{I}_{3\times3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1/T_{gs}\boldsymbol{I}_{3\times3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1/T_{as}\boldsymbol{I}_{3\times3} \end{bmatrix}$$

状态转移矩阵

$$\underset{21\times18}{\boldsymbol{G}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \boldsymbol{C}_b^n & 0 & 0 & 0 & 0 & 0 \\ 0 & \boldsymbol{C}_b^n & 0 & 0 & 0 & 0 \\ 0 & 0 & \boldsymbol{I}_{3\times3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \boldsymbol{I}_{3\times3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \boldsymbol{I}_{3\times3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \boldsymbol{I}_{3\times3} \end{bmatrix}$$

噪声驱动阵

$$\boldsymbol{\Phi}_{k/k-1} = \boldsymbol{I} + \boldsymbol{F}_{k-1}\Delta t_k, \quad \boldsymbol{q}_t \leftarrow \boldsymbol{w}_t$$

$$\boldsymbol{Q}_k = \left(\begin{matrix}\boldsymbol{\Phi}_{k/k-1}\boldsymbol{G}_{k-1}\boldsymbol{q}_{k-1}\boldsymbol{G}_{k-1}^T\boldsymbol{\Phi}_{k/k-1}^T \\ +\boldsymbol{G}_k\boldsymbol{q}_k\boldsymbol{G}_k^T\end{matrix}\right)\Delta t_k/2$$

$$\boldsymbol{x}_{k/k-1} = \boldsymbol{\Phi}_{k/k-1}\boldsymbol{x}_{k-1}$$

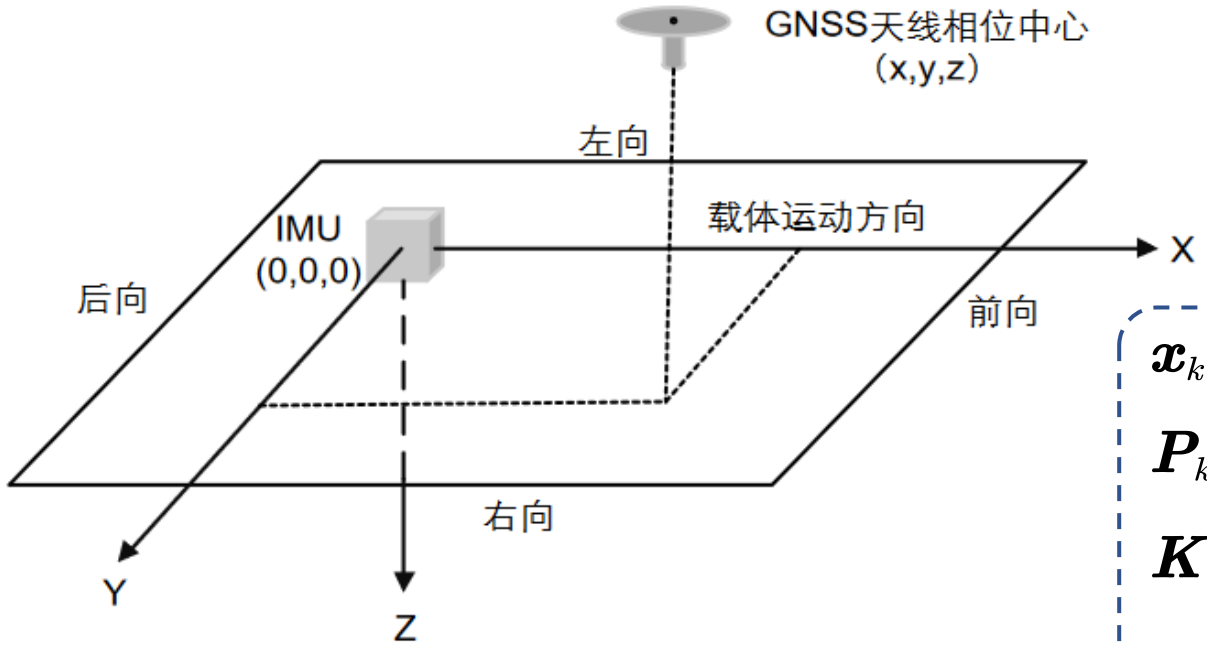
$$\boldsymbol{P}_{k/k-1} = \boldsymbol{\Phi}_{k/k-1}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}_{k/k-1}^T + \boldsymbol{Q}_k$$

系统状态预测

GNSS/INS组合导航

GNSS位置观测方程

- n系构建位置观测方程
- 位置观测向量：IMU预测天线位置减去GNSS观测位置
- 得到观测向量z，观测矩阵H，观测噪声矩阵R



$$\hat{\boldsymbol{r}}_G = \hat{\boldsymbol{r}}_I + \boldsymbol{D}_R^{-1}\hat{\boldsymbol{C}}_b^n \boldsymbol{l}^b$$

$$\boldsymbol{z}_r = \boldsymbol{D}_R(\hat{\boldsymbol{r}}_G - \tilde{\boldsymbol{r}}_G)$$

$$\boldsymbol{z}_r \approx \delta \boldsymbol{r}^n + (\boldsymbol{C}_b^n \boldsymbol{l}^b)\times \boldsymbol{\phi} + \boldsymbol{n}_r$$

$$\boldsymbol{H}_r = [\boldsymbol{I}_3 \quad \boldsymbol{0}_3 \quad (\boldsymbol{C}_b^n \boldsymbol{l}^b)\times \quad \boldsymbol{0}_3 \quad \boldsymbol{0}_3 \quad \boldsymbol{0}_3 \quad \boldsymbol{0}_3]$$

$$\boldsymbol{R}_r \leftarrow \boldsymbol{n}_r$$

构建位置观测信息

$$\boldsymbol{x}_k = \boldsymbol{x}_{k/k-1} + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}_k \boldsymbol{x}_{k/k-1})$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k)\boldsymbol{P}_{k/k-1}(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k)^T + \boldsymbol{K}_k \boldsymbol{R}_k \boldsymbol{K}_k^T$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{k/k-1}\boldsymbol{H}_k^T(\boldsymbol{H}_k \boldsymbol{P}_{k/k-1}\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}$$

卡尔曼滤波状态更新

GNSS/INS组合导航

IMU误差补偿

- 惯性导航解算之前补偿当前的零偏、比例因子
- 卡尔曼滤波估计零偏、比例因子残差

状态误差反馈

- 零偏、比例因子残差加到当前零偏、比例因子上
- 估计的误差状态反馈到导航状态并输出

更新时间判断

- GNSS整秒采样，IMU单独采样，采样时刻不对齐
- 对齐误差偏差较大时，对**IMU数据内插**

误差补偿 $diag(\boldsymbol{I}+\bar{\boldsymbol{s}}_g)^{-1}(\tilde{\boldsymbol{\omega}}_{ib}^b-\bar{\boldsymbol{b}}_g)=\hat{\boldsymbol{\omega}}_{ib}^b$

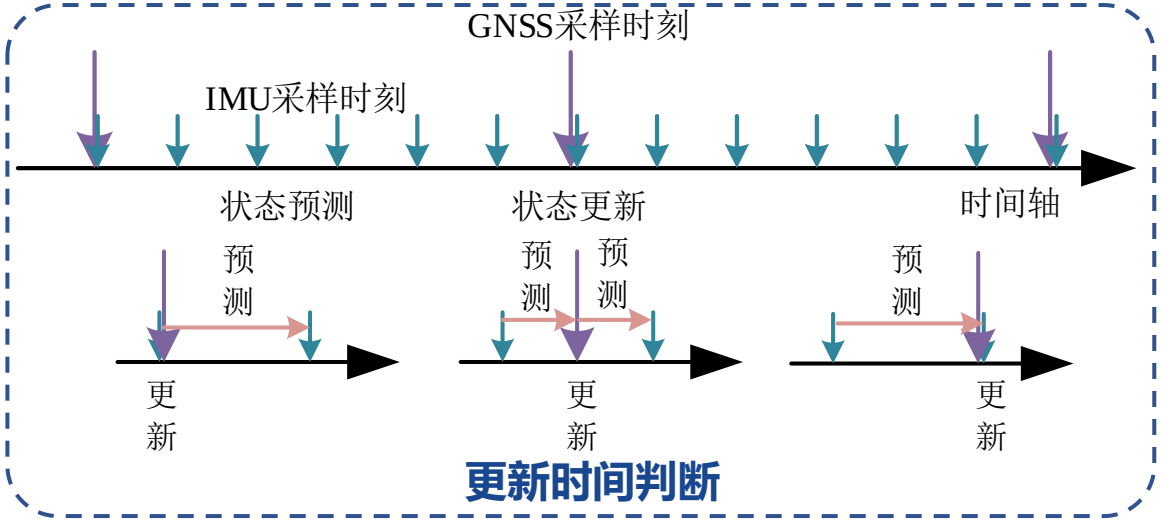
误差更新 $\bar{\boldsymbol{b}}_g \leftarrow \bar{\boldsymbol{b}}_g+\boldsymbol{b}_g,\ \bar{\boldsymbol{s}}_g \leftarrow \bar{\boldsymbol{s}}_g+\boldsymbol{s}_g$

$$\hat{\boldsymbol{r}}^n=\boldsymbol{r}^n+\delta\boldsymbol{r}^n \qquad \boldsymbol{r}=\hat{\boldsymbol{r}}-\boldsymbol{D}_r\delta\boldsymbol{r}^n$$

$$\hat{\boldsymbol{v}}^n=\boldsymbol{v}^n+\delta\boldsymbol{v}^n \quad \Rightarrow \quad \boldsymbol{v}^n=\hat{\boldsymbol{v}}^n-\delta\boldsymbol{v}^n$$

$$\hat{\boldsymbol{C}}_b^n=[\boldsymbol{I}-(\boldsymbol{\phi}\times)]\boldsymbol{C}_b^n \qquad \boldsymbol{C}_b^n=[\boldsymbol{I}+(\boldsymbol{\phi}\times)]\hat{\boldsymbol{C}}_b^n$$

误差扰动 **误差反馈**



更新时间判断

目 录

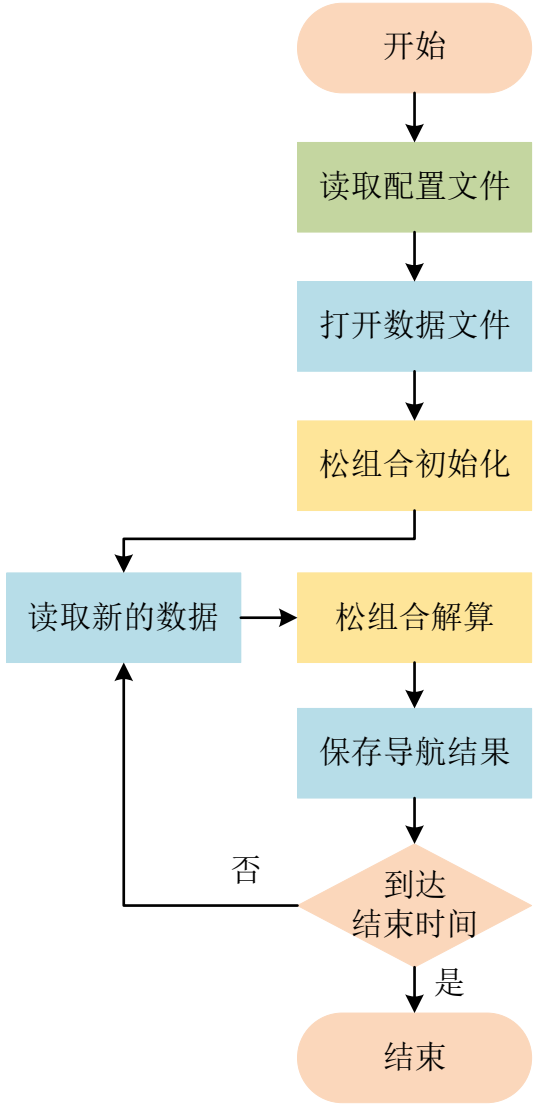
KF-GINS代码设计

软件接口需求

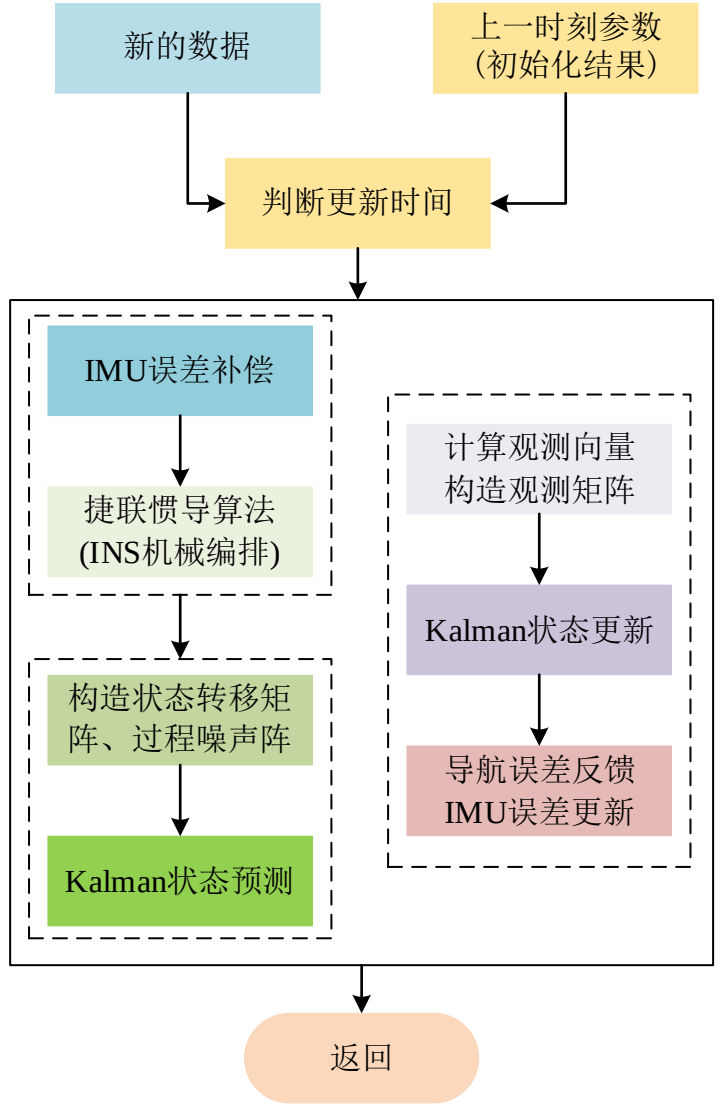
- 面向**后处理**应用，循环**读取数据文件**，逐历元保存结果
- 配置文件用来**配置程序参数**

松组合算法模块

- 松组合算法初始化
- 更新时间判断
- IMU误差补偿
- 惯性导航解算
- 状态转移矩阵和噪声驱动阵构建
- GNSS位置观测构建
- 卡尔曼滤波状态预测和状态更新
- 误差状态反馈



KF-GINS代码主流程

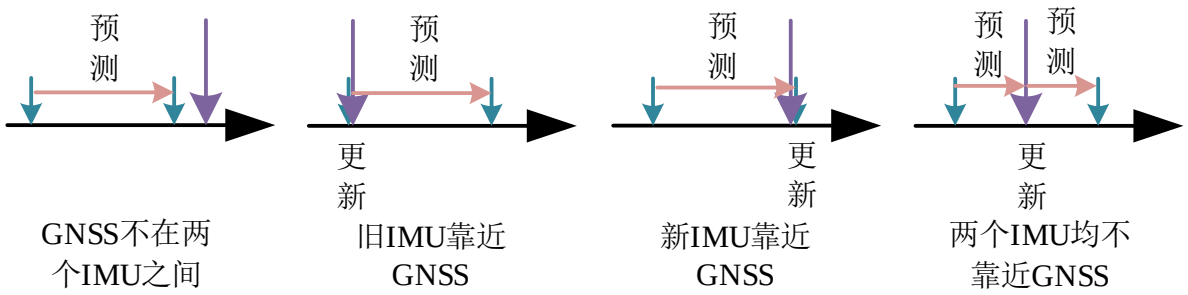


松组合解算内容

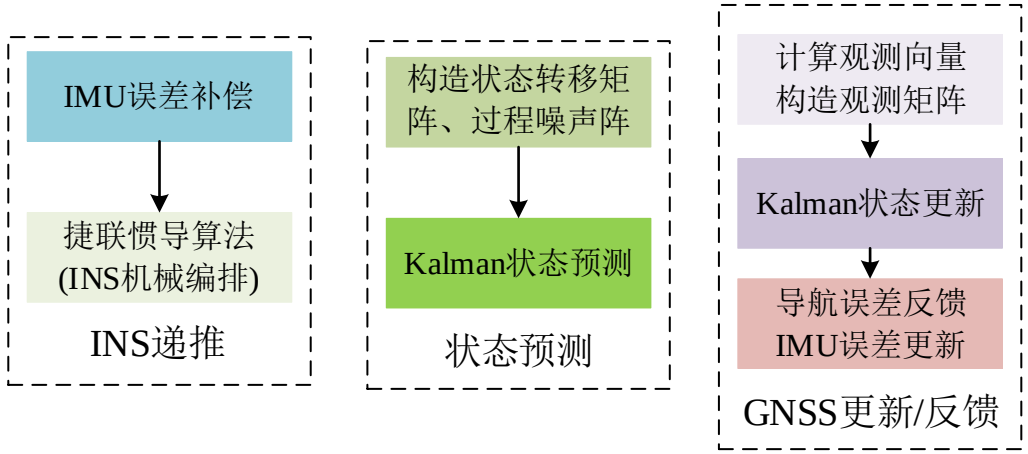
KF-GINS代码设计

松组合处理流程

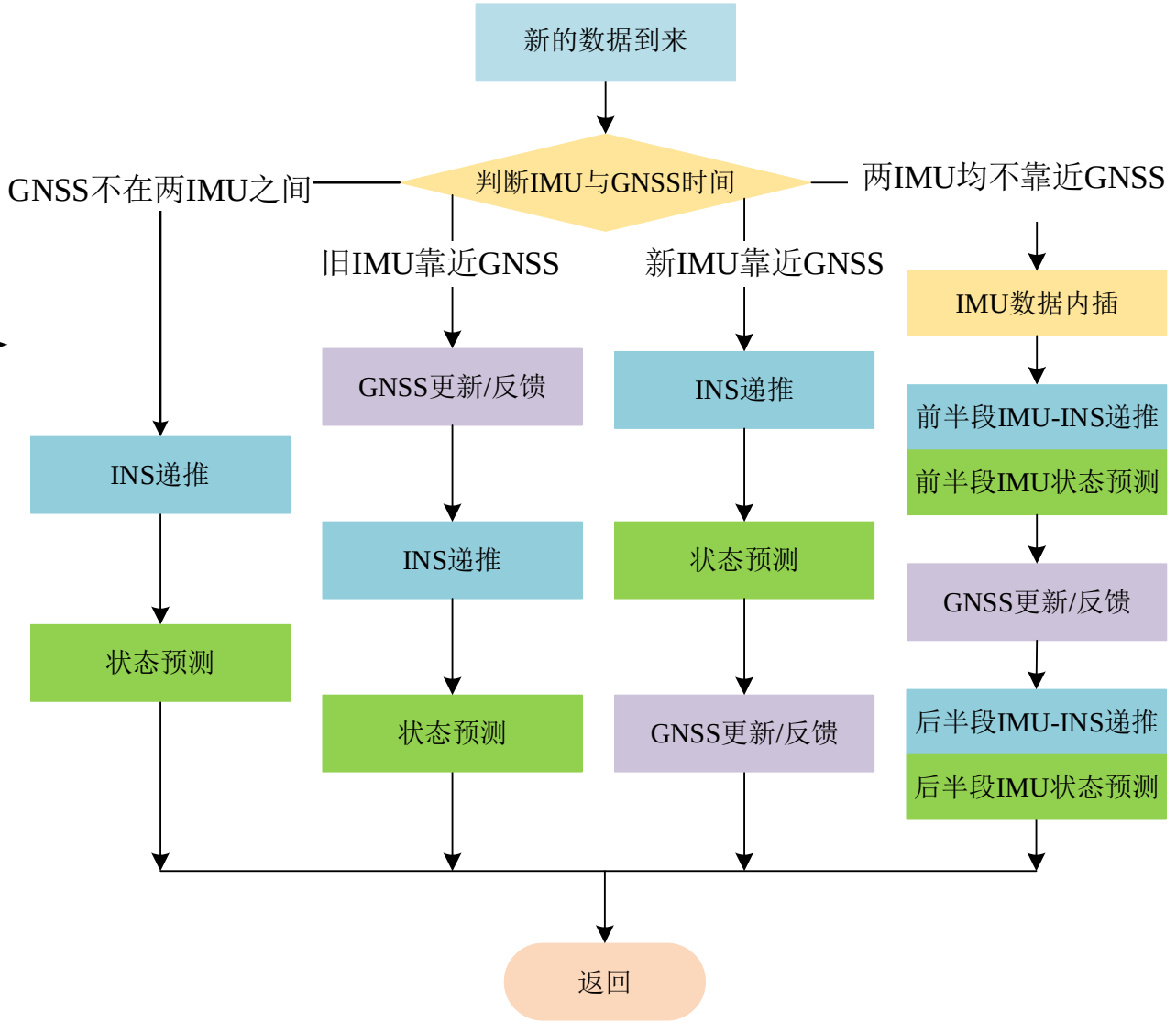
- 根据IMU和GNSS时间判断处理机制



IMU和GNSS时间情况



松组合功能模块

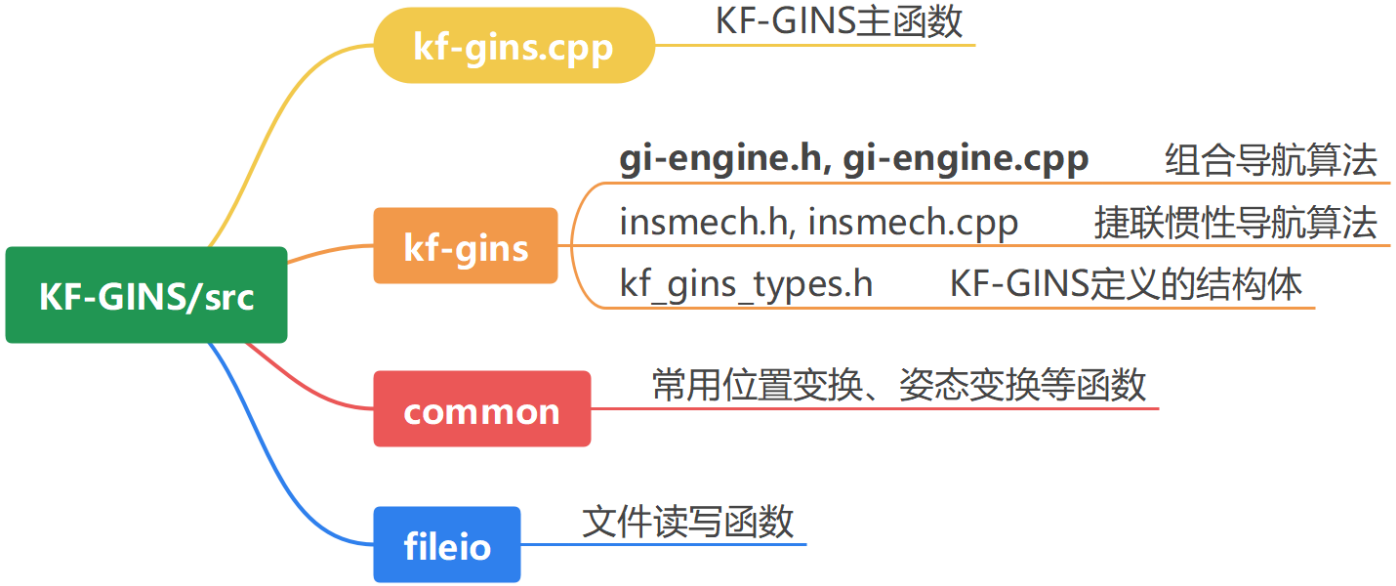


松组合处理机制和流程

KF-GINS代码结构

KF-GINS源代码结构

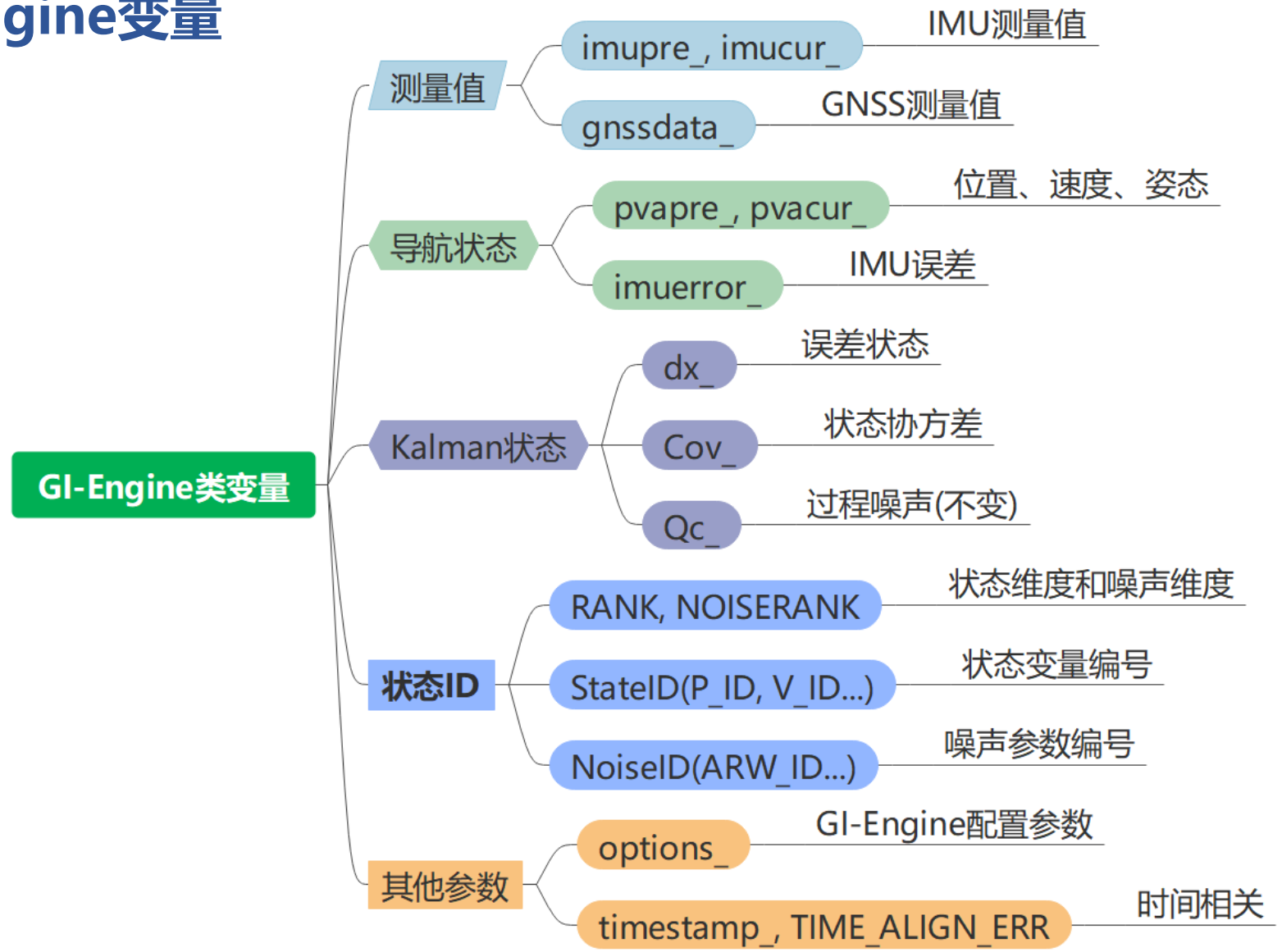
- KF-GINS主函数：**kf_gins.cpp**，程序入口，读取数据，调用组合算法
- 组合导航算法类：**GIEngine，组合导航算法变量和实现函数**
- 惯性导航算法类：**INSMech**，速度更新、位置更新、姿态更新
- KF-GINS定义结构体：**kf_gins_types.h**，配置文件结构体
- 常用转换函数类：Rotation等；文件读写类：FileLoder等(沿用OB-GINS中的类)



KF-GINS源代码文件

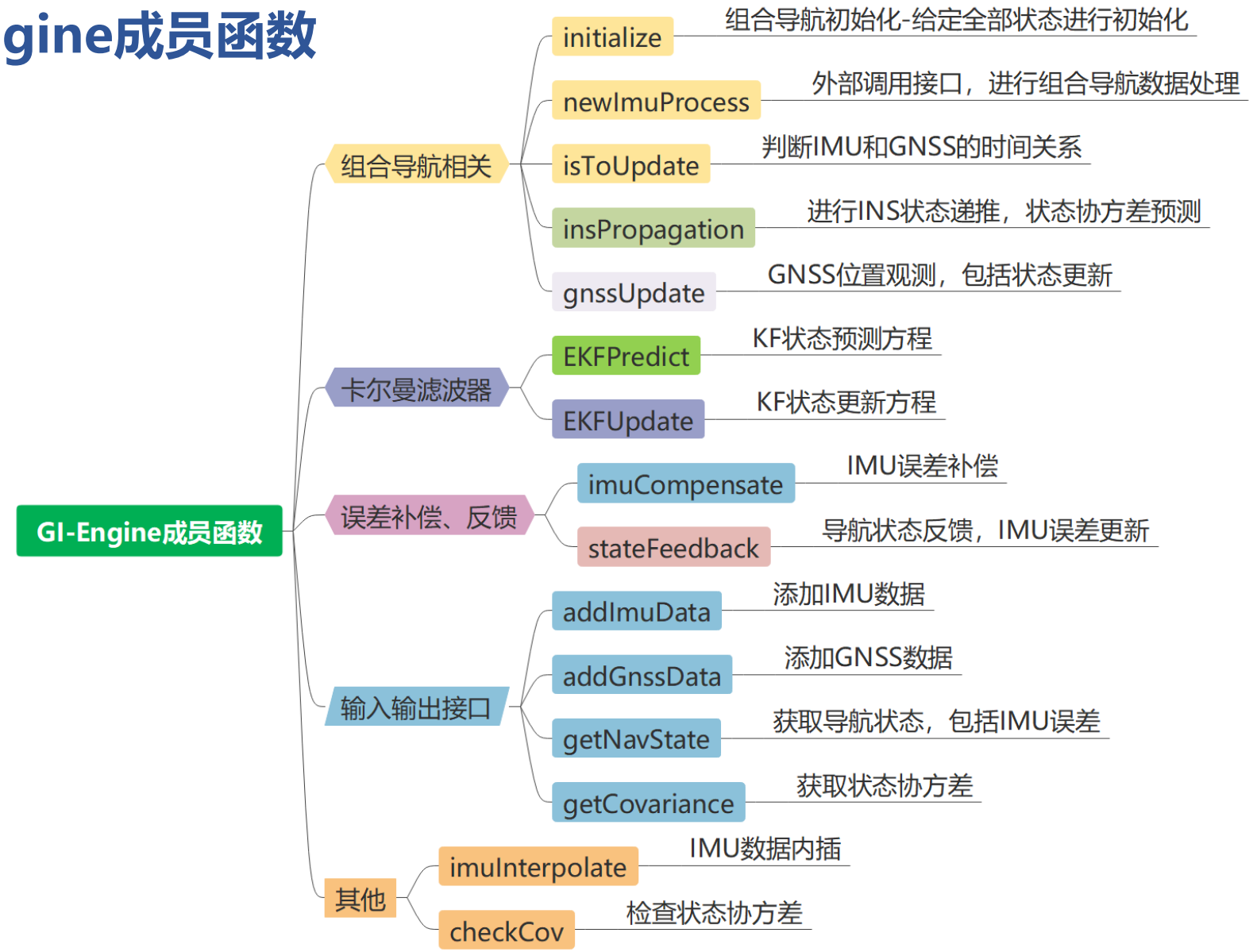
KF-GINS代码结构

核心类GI-Engine变量



KF-GINS代码结构

核心类GI-Engine成员函数



代码实现-惯导速度更新

$$\boldsymbol{v}_{eb,k}^{n} = \boldsymbol{v}_{eb,k-1}^{n} + \underbrace{\left[\boldsymbol{I} - \frac{1}{2}(\boldsymbol{\zeta}_{n(k-1)n(k)}\times)\right]\boldsymbol{C}_{b(k-1)}^{n(k-1)}\Delta\boldsymbol{v}_{f,k}^{b(k-1)}}_{n\text{系比力积分项}}$$

$$+ \underbrace{\boldsymbol{g}_{l}^{n}\Delta t_k - (2\boldsymbol{\omega}_{ie}^{n} + \boldsymbol{\omega}_{en}^{n})\times\boldsymbol{v}_{eb}^{n}\big|_{t=t_{k-1/2}}\Delta t_k}_{\text{重力/哥氏积分项}}$$

捷联惯导速度更新

$$\boldsymbol{\zeta}_{n(k-1)n(k)} \approx \boldsymbol{\omega}_{in}^{n}\big|_{t=t_{k-1/2}}\Delta t_k$$

$$\underbrace{\Delta\boldsymbol{v}_{f,k}^{b(k-1)}}_{b\text{系比力积分项}} = \Delta\boldsymbol{v}_k + \frac{1}{2}\Delta\boldsymbol{\theta}_k\times\Delta\boldsymbol{v}_k + \frac{1}{12}(\Delta\boldsymbol{\theta}_{k-1}\times\Delta\boldsymbol{v}_k + \Delta\boldsymbol{v}_{k-1}\times\Delta\boldsymbol{\theta}_k)$$

哥氏积分项
划桨、旋转效应

速度更新

- 计算b系比力积分项和n系下的比力积分项
- k-1时刻状态计算重力/哥氏积分项
- 更新得到k时刻速度，位置
- 计算**k-1/2时刻**的位置
- k-1/2时刻状态重新计算n系比力积分项
- **重新计算**重力/哥氏积分项
- 更新得到k时刻的速度

kf-gins/insmech.cpp/INSMech::velUpdate

```
// b系比力积分项,旋转效应和划桨效应
temp1 = imucur.dtheta.cross(imucur.dvel) / 2;
temp2 = imupre.dtheta.cross(imucur.dvel) / 12;
temp3 = imupre.dvel.cross(imucur.dtheta) / 12;
d_vfb = imucur.dvel + temp1 + temp2 + temp3;
// 比力积分项投影到n系
temp3 = (wie_n + wen_n) * imucur.dt / 2;
cnn   = I33 - Rotation::skewSymmetric(temp3);
d_vfn = cnn * pvapre.att.cbn * d_vfb;

// 计算重力/哥式积分项
d_vgn = (gl - (2 * wie_n +
wen_n).cross(pvapre.vel)) * imucur.dt;

// 外推得到中间时刻位置
midpos = Earth::blh(qne, midpos[2]);

// 重新计算n系下平均比力积分项
d_vfn = cnn * pvapre.att.cbn * d_vfb;
// 重新计算重力、哥式积分项
d_vgn = (gl - (2 * wie_n + wen_n).cross(midvel))
* imucur.dt;

// 速度更新完成
pvacur.vel = pvapre.vel + d_vfn + d_vgn;
```

代码实现-惯导位置/姿态更新

捷联惯导位置更新

$$h_k = h_{k-1} - \frac{1}{2}(v_{D,k-1} + v_{D,k})\Delta t_k$$

$$\varphi_k = \varphi_{k-1} + \frac{v_{N,k-1} + v_{N,k}}{2(R_{M,k-\frac{1}{2}} + h_{k-\frac{1}{2}})}\Delta t_k$$

$$\lambda_k = \lambda_{k-1} + \frac{v_{E,k-1} + v_{E,k}}{2(R_{N,k-\frac{1}{2}} + h_{k-\frac{1}{2}})\cos\varphi_{k-\frac{1}{2}}}\Delta t_k$$

位置更新

- 更新高程，计算平均高程
- 更新纬度，计算平均纬度
- 更新经度
- 得到k时刻的位置

姿态更新

- 计算k-1/2时刻的位置和速度
- 计算**n系旋转四元数**
- 双子样假设计算**b系旋转四元数**
- 更新k时刻姿态

等效旋转矢量更新圆锥效应

$$\boldsymbol{q}_{b_k}^{n_k} = \boldsymbol{q}_{n_{k-1}}^{n_k}\boldsymbol{q}_{b_{k-1}}^{n_{k-1}}\boldsymbol{q}_{b_k}^{b_{k-1}}$$

$$\boldsymbol{q}_{n_{k-1}}^{n_k} \leftarrow \boldsymbol{\zeta}_k \approx \boldsymbol{\omega}_{in}^{n}\big|_{t=t_{k-1/2}}\Delta t_k$$

捷联惯导姿态更新

$$\boldsymbol{q}_{b_k}^{b_{k-1}} \leftarrow \boldsymbol{\phi}_k = \Delta\boldsymbol{\theta}_k + \frac{1}{12}\Delta\boldsymbol{\theta}_{k-1}\times\Delta\boldsymbol{\theta}_k$$

kf-gins/insmech.cpp/INSMech::attUpdate

```
// 重新计算中间时刻的速度和位置，地理参数
midpos    = Earth::blh(qne_mid, midpos[2]);

// 计算n系的旋转四元数 k-1时刻到k时刻变换
temp1 = -(wie_n + wen_n) * imucur.dt;
qnn   = Rotation::rotvec2quaternion(temp1);

// 计算b系旋转四元数 补偿二阶圆锥误差
temp1 = imucur.dtheta + imupre.dtheta.cross(imucur.dtheta) / 12;
qbb   = Rotation::rotvec2quaternion(temp1);

// 姿态更新完成pvacur.att.qbn    = qnn * pvapre.att.qbn * qbb;
pvacur.att.cbn   = Rotation::quaternion2matrix(pvacur.att.qbn);
pvacur.att.euler = Rotation::matrix2euler(pvacur.att.cbn);
```

E.-H. Shin, “Estimation techniques for low-cost inertial navigation,” Cal, 2005. (位置更新方程参考)

代码实现-组合导航

误差状态预测

- 调用惯性导航算法(机械编排算法)
- 构造**噪声驱动阵**，计算过程噪声方差
- Kalman方程进行**协方差传播**和**状态预测**

状态转移矩阵、过程噪声

$$\boldsymbol{\Phi}_{k/k-1} = \boldsymbol{I} + \boldsymbol{F}_{k-1}\Delta t_k$$

$$\boldsymbol{Q}_k = \begin{pmatrix} \boldsymbol{\Phi}_{k/k-1}\boldsymbol{G}_{k-1}\boldsymbol{q}_{k-1}\boldsymbol{G}_{k-1}^T\boldsymbol{\Phi}_{k/k-1}^T \\ +\boldsymbol{G}_k\boldsymbol{q}_k\boldsymbol{G}_k^T \end{pmatrix}\Delta t_k/2$$

状态、协方差传播

$$\boldsymbol{x}_{k/k-1} = \boldsymbol{\Phi}_{k/k-1}\boldsymbol{x}_{k-1}$$

$$\boldsymbol{P}_{k/k-1} = \boldsymbol{\Phi}_{k/k-1}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}_{k/k-1}^T + \boldsymbol{Q}_k$$

kf-gins/gi_engine.cpp/GIEngine::insPropagation

```
// 对当前IMU数据(imucur)补偿误差
imuCompensate(imucur);
// IMU状态更新(机械编排算法)
INSMech::insMech(pvapre_, pvacur_, imupre, imucur);
... ... ...
F.block(P_ID, P_ID, 3, 3) = temp;
F.block(P_ID, V_ID, 3, 3) = Eigen::Matrix3d::Identity();
F.block(V_ID, V_ID, 3, 3)   = temp;
F.block(V_ID, PHI_ID, 3, 3) =
Rotation::skewSymmetric(pvapre_.att.cbn * accel);
F.block(V_ID, BA_ID, 3, 3)  = pvapre_.att.cbn;
F.block(V_ID, SA_ID, 3, 3)  = pvapre_.att.cbn *
(accel.asDiagonal());
... ... ...
// 状态转移矩阵
Phi.setIdentity();
Phi = Phi + F * imucur.dt;
// 计算系统传播噪声
Qd = G * Qc_ * G.transpose() * imucur.dt;
Qd = (Phi * Qd * Phi.transpose() + Qd) / 2;
```

kf-gins/gi_engine.cpp/GIEngine::EKFPredict

```
// 传播系统协方差和误差状态
Cov_ = Phi * Cov_ * Phi.transpose() + Qd;
dx_  = Phi * dx_;
```

代码实现-组合导航

GNSS位置更新

- 构建GNSS位置观测
- Kalman方程进行状态更新

$$\hat{\boldsymbol{r}}_G = \hat{\boldsymbol{r}}_I + \boldsymbol{D}_R^{-1}\hat{\boldsymbol{C}}_b^n \boldsymbol{l}^b$$

$$\boldsymbol{z}_r = \boldsymbol{D}_R(\hat{\boldsymbol{r}}_G - \tilde{\boldsymbol{r}}_G)$$

$$\boldsymbol{z}_r \approx \delta \boldsymbol{r}^n + (\boldsymbol{C}_b^n \boldsymbol{l}^b)\times \boldsymbol{\phi} + \boldsymbol{n}_r$$

$$\boldsymbol{H}_r = [\boldsymbol{I}_3 \quad \boldsymbol{0}_3 \quad (\boldsymbol{C}_b^n \boldsymbol{l}^b)\times \quad \boldsymbol{0}_3 \quad \boldsymbol{0}_3 \quad \boldsymbol{0}_3 \quad \boldsymbol{0}_3]$$

$$\boldsymbol{R}_r \leftarrow \boldsymbol{n}_r$$

构建位置观测信息

kf-gins/gi_engine.cpp/GIEngine:gnssUpdate

```
// GNSS位置观测向量
dz = Dr * (antenna_pos - gnssdata.blh);

// 构造GNSS位置观测矩阵
H_gnsspos.setZero();
H_gnsspos.block(0, P_ID, 3, 3) = Eigen::Matrix3d::Identity();
H_gnsspos.block(0, PHI_ID, 3, 3) =
Rotation::skewSymmetric(pvacur_.att.cbn * options_.antlever);

// 位置观测噪声阵
R_gnsspos =
gnssdata.std.cwiseProduct(gnssdata.std).asDiagonal();
```

kf-gins/gi_engine.cpp/GIEngine::EKFUpdate

```
// 计算Kalman增益
temp = H * Cov_ * H.transpose() + R;
K = Cov_ * H.transpose() * temp.inverse();
// 更新系统误差状态和协方差
I = I - K * H;
dx_  = dx_ + K * (dz - H * dx_);
Cov_ = I * Cov_ * I.transpose() + K * R * K.transpose();
```

代码实现-组合导航

IMU误差补偿

- 捷联惯性导航算法之前进行误差补偿
- 补偿当前估计的零偏和比例因子

$$diag\left(\boldsymbol{I}+\bar{\boldsymbol{s}}_g\right)^{-1}\left(\tilde{\boldsymbol{\omega}}_{ib}^b-\bar{\boldsymbol{b}}_g\right)=\hat{\boldsymbol{\omega}}_{ib}^b$$

$$diag\left(\boldsymbol{I}+\bar{\boldsymbol{s}}_a\right)^{-1}\left(\tilde{\boldsymbol{f}}_{ib}^b-\bar{\boldsymbol{b}}_a\right)=\hat{\boldsymbol{f}}_{ib}^b$$

导航误差反馈和IMU误差更新

- 按照扰动形式进行导航误差反馈
- 估计的IMU残差更新到IMU误差上
- **误差反馈后，系统状态置零**

$$\boldsymbol{r}=\hat{\boldsymbol{r}}-\boldsymbol{D}_r\delta\boldsymbol{r}^n \qquad \bar{\boldsymbol{b}}_g\leftarrow\bar{\boldsymbol{b}}_g+\boldsymbol{b}_g,\ \bar{\boldsymbol{b}}_a\leftarrow\bar{\boldsymbol{b}}_a+\boldsymbol{b}_a$$

$$\boldsymbol{v}^n=\hat{\boldsymbol{v}}^n-\delta\boldsymbol{v}^n \qquad \bar{\boldsymbol{s}}_g\leftarrow\bar{\boldsymbol{s}}_g+\boldsymbol{s}_g,\bar{\boldsymbol{s}}_a\leftarrow\bar{\boldsymbol{s}}_a+\boldsymbol{s}_a$$

$$\boldsymbol{C}_b^n=[\boldsymbol{I}+(\boldsymbol{\phi}\times)]\hat{\boldsymbol{C}}_b^n$$

kf-gins/gi_engine.cpp/GIEngine:imuCompensate

```
// 补偿IMU零偏
imu.dtheta -= imuerror_.gyrbias * imu.dt;
imu.dvel -= imuerror_.accbias * imu.dt;
// 补偿IMU比例因子
gyrscale = Eigen::Vector3d::Ones() + imuerror_.gyrscale;
accscale = Eigen::Vector3d::Ones() + imuerror_.accscale;
imu.dtheta = imu.dtheta.cwiseProduct(gyrscale.cwiseInverse());
imu.dvel = imu.dvel.cwiseProduct(accscale.cwiseInverse());
```

kf-gins/gi_engine.cpp/GIEngine::stateFeedback

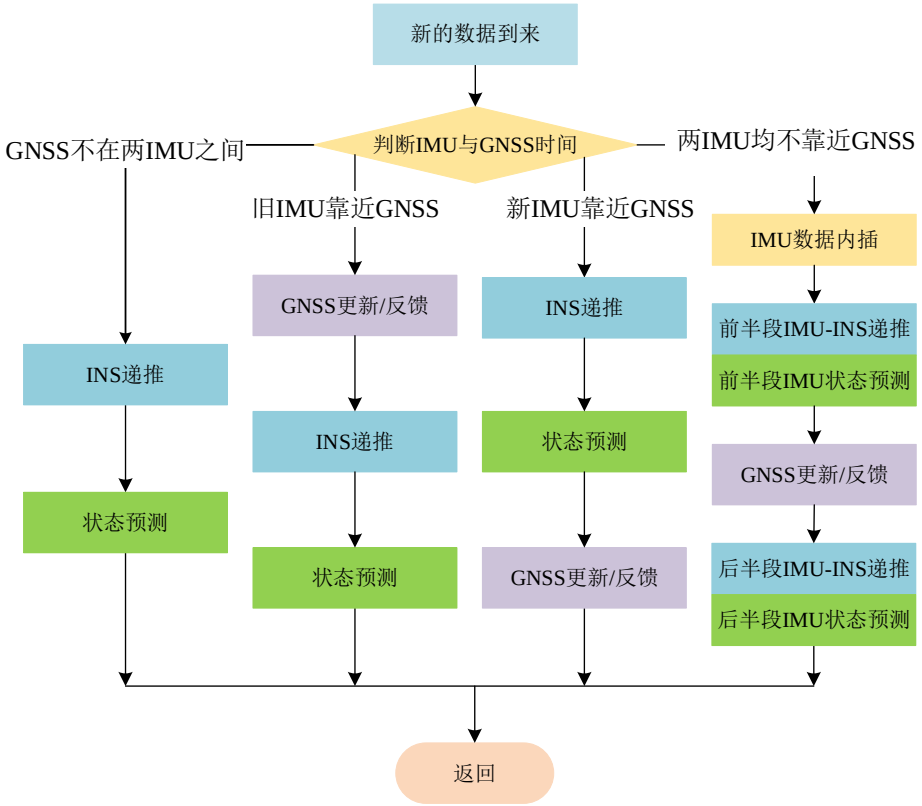
```
// 位置、速度误差反馈
Eigen::Vector3d delta_r = dx_.block(P_ID, 0, 3, 1);
pvacur_.pos -= Dr_inv * delta_r;
vectemp = dx_.block(V_ID, 0, 3, 1);
pvacur_.vel -= vectemp;
// 姿态误差反馈
vectemp                 = dx_.block(PHI_ID, 0, 3, 1);
Eigen::Quaterniond qpn = Rotation::rotvec2quaternion(vectemp);
pvacur_.att.qbn         = qpn * pvacur_.att.qbn;
// IMU零偏误差反馈
vectemp = dx_.block(BA_ID, 0, 3, 1);
imuerror_.accbias += vectemp;

// 误差状态反馈到系统状态后,将误差状态清零
dx_.setZero();
```

代码实现-组合导航

IMU数据处理

- 读取到新数据后**判断IMU和GNSS时间**
- 根据不同的情况，进入**不同的处理机制**
- 完成组合导航算法主流程



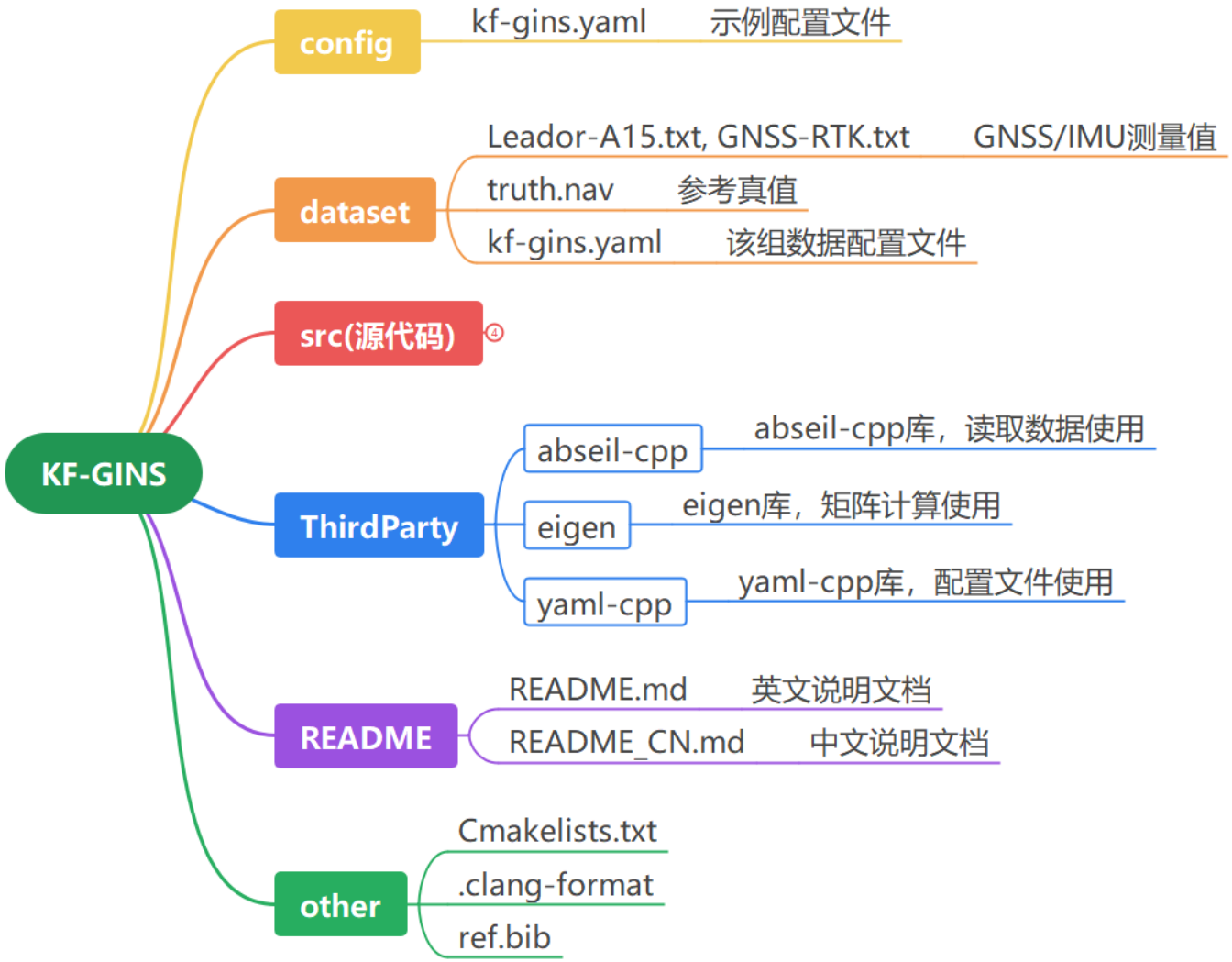
kf-gins/gi_engine.cpp/GIEngine:newImuProcess

```
// 判断是否需要进行GNSS更新
int res = isToUpdate(imupre_.time, imucur_.time, updatetime);
// 只传播导航状态
if (res == 0) { insPropagation(imupre_, imucur_); }
else if (res == 1) {
    // GNSS数据靠近上一历元，先对上一历元进行GNSS更新
    gnssUpdate(gnssdata_);
    stateFeedback();
    pvapre_ = pvacur_;  insPropagation(imupre_, imucur_); }
else if (res == 2) {
    // GNSS数据靠近当前历元，先对当前IMU进行状态传播
    insPropagation(imupre_, imucur_);
    gnssUpdate(gnssdata_);
    stateFeedback();  }
else {
    // GNSS数据不靠近任何一个，将当前IMU内插到整秒时刻
    imuInterpolate(imupre_, imucur_, updatetime, midimu);
    // 对前一半IMU进行状态传播
    insPropagation(imupre_, midimu);
    // 整秒时刻进行GNSS更新，并反馈系统状态
    gnssUpdate(gnssdata_);
    stateFeedback();
    // 对后一半IMU进行状态传播
    pvapre_ = pvacur_;   insPropagation(midimu, imucur_); }
// 检查协方差矩阵对角线元素
checkCov();
```

KF-GINS工程总览

KF-GINS工程总览

- **后处理**GNSS/INS**松**组合算法
- 读取IMU、GNSS数据，输出**导航结果、IMU误差、状态标准差**
- **配置文件**设置程序参数
- C++编写，Cmake管理工程
- 包含程序依赖的三方库
- 多平台运行(建议**首选linux**)
- 提供**测试数据**和对应配置文件
- 说明文档介绍程序配置、文件格式等



KF-GINS项目文件

KF-GINS程序运行

KF-GINS程序运行

- 安装编译需要的库，编译并运行测试数据
- Dataset文件夹生成KF_GINS_Navresult.txt, KF_GINS_IMU_ERR.txt, KF_GINS_STD.txt

```
# 安装编译需要的库
sudo apt-get install cmake
sudo apt-get install build-essential
# 克隆KF-GINS到本地
git clone https://github.com/i2Nav-WHU/KF-GINS.git ~/

# 编译 KF-GINS
cd ~/KF-GINS
mkdir build && cd build
cmake ../ -DCMAKE_BUILD_TYPE=Release
make -j8

# 运行测试数据
cd ~/KF-GINS
./bin/KF-GINS ./dataset/kf-gins.yaml
```

KF-GINS下载和编译

```
(base) ➜ KF-GINS git:(main) ✗ ./bin/KF-GINS ./dataset/kf-gins.yaml

KF-GINS: An EKF-Based GNSS/INS Integrated Navigation System

---------------KF-GINS Options:---------------
 - Initial State:
        - initial position: 30.4447873701  114.471863205  20.899 [deg, deg, m]
        - initial velocity: 0 0 0 [m/s]
        - initial attitude: 0.854215  -2.0348  185.702 [deg]
        - initial gyrbias : 0 0 0 [deg/h]
        - initial accbias : 0 0 0 [mGal]
        - initial gyrscale: 0 0 0 [ppm]
        - initial accscale: 0 0 0 [ppm]
 - Initial State STD:
        - initial position std: 0.005 0.004 0.008 [m]
        - initial velocity std: 0.003 0.004 0.004 [m/s]
        - initial attitude std: 0.003 0.003 0.023 [deg]
        - initial gyrbias std: 0.027 0.027 0.027 [deg/h]
        - initial accbias std: 15 15 15 [mGal]
        - initial gyrscale std: 300 300 300 [ppm]
        - initial accscale std: 300 300 300 [ppm]
 - IMU noise:
        - arw: 0.003 0.003 0.003 [deg/s/sqrt(h)]
        - vrw: 0.03 0.03 0.03 [m/s/sqrt(h)]
        - gyrbias  std: 0.027 0.027 0.027 [deg/h]
        - accbias  std: 15 15 15 [mGal]
        - gyrscale std: 300 300 300 [ppm]
        - accscale std: 300 300 300 [ppm]
        - correlation time: 4 [h]
 - Antenna leverarm:  0.136 -0.301 -0.184 [m]

 - Processing: 100%

KF-GINS Process Finish! From 456300 s to 459665 s, total 3364.62 s!
Cost 20.7331 s in total
```

KF-GINS程序运行

目 录

KF-GINS常见问题

- KF-GINS能够达到怎么样的定位精度?

  组合导航算法精度**更受IMU等级、以及测试时GNSS定位质量**影响。组合导航算法相对成熟，对于同样的设备只要算法正确实现，算法几乎不会对定位精度产生较大影响

- 初始导航状态和初始导航状态标准差如何给定?

  初始位置(和速度)可由**GNSS给定**，初始姿态(和速度)可从**参考结果中获取**;

  位置速度标准差可由GNSS给定，姿态标准差可给经验值，车载领域一般横滚俯仰小，航向大一些

- IMU数据输入到程序之前，需要扣除重力加速度吗?

  不需要，**惯性导航算法中补偿了重力加速度**

- INS机械编排中旋转效应等补偿项，对于低端IMU是否需要补偿?

  低端IMU测量噪声更大，**简化的IMU积分算法对低端IMU精度产生的影响较小**

  参考：Zhang Q, Niu X, Zhang H, et al. Algorithm improvement of the low-end GNSS/INS systems for land vehicles navigation[J]. Mathematical Problems in Engineering, 2013, 2013

KF-GINS常见问题

- 组合导航中GNSS信号丢失期间进行纯惯导解算，这时IMU误差项可以补偿吗？

  GNSS丢失期间IMU误差项不更新，但是可以**利用之前估计的IMU误差项继续补偿**

- IMU数据，如何从速率形式转到增量形式？

  一般采用更高频率速率数据积分得到增量数据，参考：新手入门系列4——MEMS IMU**原始数据采集和时间同步**的那些坑(i2Nav网站)

- IMU零偏和比例因子建模时相关时间如何给定？

  建模为一阶高斯-马尔可夫过程，实际中一般**根据经验设置相关时间，MEMS设置1h**

- GNSS/INS组合导航中是否需要考虑惯性系和车体系的转换？

  不需要，GNSS/INS组合导航不受载体限制，不需要考虑车体系

KF-GINS拓展

初始化拓展

- 动态初始对准，GNSS位置差分速度(或者GNSS直接输出速度)，位置差分计算初始航向
- **快速航向初始化**[1]，轨迹匹配方法快速获取准确初始航向
- 静态粗对准，适用于高精度惯导，双矢量法找到初始姿态

观测信息拓展

- 直接构建观测向量、观测模型和噪声矩阵，调用EKFUpdate更新和stateFeedback反馈
- GNSS速度观测信息、NHC约束信息(对于车载)、零速信息修正

状态信息拓展

- 如果需要增广系统状态(如里程计增广比例因子[2])，则修改RANK(NOISERANK)，添加StateID, NoiseID
- 协方差、状态转移矩阵、观测信息对应修改；添加观测信息，进行更新反馈

[1] Q. Chen, H. Lin, J. Kuang, Y. Luo, and X. Niu, "**Rapid Initial Heading Alignment for MEMS Land Vehicular GNSS/INS Navigation System**," *IEEE Sensors J.*, vol. 23, no. 7, pp. 7656–7666, Apr. 2023, doi: 10.1109/JSEN.2023.3247587.
[2] L. Wang, X. Niu, T. Zhang, H. Tang, and Q. Chen, "**Accuracy and robustness of ODO/NHC measurement models for wheeled robot positioning**," *Measurement*, vol. 201, p. 111720, Sep. 2022, doi: 10.1016/j.measurement.2022.111720.

欢迎指导交流！

王立强、陈起金

武汉大学多源智能导航实验室（i2Nav）

www.i2nav.com

https://github.com/i2Nav-WHU

2023年5月14日

i2Nav微信公众号

i2Nav开源算法交流QQ群