雾膜软件

WMS502 代码包

标准版本惯性卫星松组合导航 C++语言

版本 20230824

1. 内容介绍

本代码包的功能主要有：根据陀螺仪和加速度计信息计算惯性导航；根据卫星定位信息计算组合导航。

采用 15 维状态量、6 维观测量的扩展卡尔曼滤波方法。

带有一组光纤陀螺仪惯性导航装置在固定翼飞机飞行实验数据。

代码包适用于：组合导航的学习、研究、教学、科研；处理组合导航实验数据。

本代码为了便于学习理解，力求简洁清楚。只保留了关键的、普适的导航计算原理。一些差异化功能没有包含在本代码中，例如：通信协议、初始对准、传感器标定、生成仿真路线等等。需要其余功能的用户可以联系本店定制开发。

2. 应用说明

2.1. 概念定义

惯性测量单元为 3 轴陀螺仪和 3 轴加速度计。定义 x 向东、y 向北、z 向天为姿态 0 位置。旋转方向和角速度方向满足右手法则，即右手握住坐标轴，大拇指位于坐标轴正向，则其余四个手指指向旋转正向。姿态的欧拉角旋转顺序定义为依次绕 z、x、y 旋转。

组合导航中，扩展卡尔曼滤波的状态变量定义为 15 维度的误差量：纬经高，东北天速度，东北天姿态，三轴陀螺仪零偏，三轴加速度计零偏。

如果无特殊说明，一般采用国际单位制。角度单位为 rad，角速度单位为 rad/s，速度单位 m/s，加速度单位 m/s/s。

2.2. 编译环境

经过检验，代码可以在 Visual Studio 2019 环境，以及 Visual Studio 2013 环境，编译成功。在其他编译环境下，可能需要稍加改动才能编译成功。

本程序需要读取文件中的数据，需要把文件放在合适的位置，否则程序可能因为读不到文件而运行异常。具体使用方法参见附带的操作视频。

2.3. 主程序

主要工作过程为 main.cpp 的 instance()函数。

程序主要工作流程为：

```
设定初值
while(1)
{
    读取数据
    惯性导航计算
    更新状态方程
    if(间隔一段时间收到卫星数据)
    {
        计算卡尔曼滤波
        根据卡尔曼滤波的计算结果补偿惯性导航的误差
    }
```

```
        保存数据
    }
```

2.4. 矩阵计算库

为了矩阵计算，C++代码中有类 MAT。其内容为：

```
        int m;//行数
        int n;//列数
        double num[MAT_MAX][MAT_MAX];//矩阵数据内容
```

可以根据需要直接修改矩阵的数值。特别注意，C 或 C++中数组元素的下标从 0 开始；而 MATLAB 的下标从 1 开始。

矩阵计算的功能已经包含在代码包中，通常情况下不需要修改。

本代码包的矩阵运算是专门简化过的、另起炉灶的计算库，与 Eigen、OpenCV 等常见矩阵库不兼容。

2.5. 设定初值

设定传感器的采样间隔时间 dTins，单位为 s。

初始姿态存储于四元数 qa 中。可以直接修改 qa 以改变初始姿态。或者可以通过 setoula 函数，用欧拉角设置姿态四元数。setoula 函数输入单位为度。

初始速度存储于 tspeed 中，可以直接修改。

初始位置存储于 tpos 中，可以直接修改。顺序为纬度、经度、高度。注意纬度和经度的单位为 rad。

卡尔曼滤波中，初始状态的方差矩阵为 Pk。测量的方差矩阵为 R。系统噪声的方差矩阵为 Q，在本代码包中，Q 与时间和 Q0 有关。可以根据需要设置上述参数。

2.6. 导航计算

惯性导航计算为 ins_gyroacc()函数。

卫星导航的处理为 sat()函数。

现有代码按照固定间隔计算卫星数据；但是在采用真实数据时，应该按照实际的数据间隔计算。

2.7. 数据格式

输入数据 datain.txt 为 12 列，3 个角速度（rad/s），3 个加速度（m/s/s），3 个卫星位置（rad, rad, m），3 个卫星速度（m/s）。

输出数据前 9 列为导航结果，存储于 dataout.txt。前 9 列为 3 个姿态角（deg）、3 个速度（m/s）、3 个位置（rad, rad, m）。注意此处姿态欧拉角单位为度。

3. 计算原理

3.1. 坐标系

载体系 b 定义为与载体固定连接的坐标系，不妨取 *xyz* 轴为右前上。

地理系 t 定义为与载体处地面重合的坐标系，不妨取 *xyz* 轴为东北天。

导航坐标系 n 是表示导航结果的坐标系。在航海、航空领域中，为了避免船只、飞机通过南北极附近时 n 系快速旋转导致导航结果异常，n 系会与 t 系有一定的夹角。在普通导航系统中，可以不虑载体通过南北极的情况，因此选取 n 系与 t 系重合以使导航算法简化。

平台坐标系 p，是平台式导航系统中传感器的指向，或者是捷联式导航系统中数学换算后的传感器的指向。理想情况下 p 系与 n 系重合；但是由于陀螺仪误差等因素，真实的 p 系与 n 系有误差角。捷联式导航系统希望把加速度换算到 n 系中，但是实际上是换算到了 p 系中。在一般的导航计算中，不必刻意区分 p 系和 n 系，但是在分析误差时需要引入 p 系。

地球坐标系 e，是和地球固连的坐标系，不妨规定 *z* 轴沿着南北极方向指向北，*x* 轴指向 0 经度方向。

惯性参考系 i。惯性参考系主要用于描述概念。惯性导航中一般不需要真正地在惯性参考系中投影，所以不必在惯性参考系中规定坐标系。

完整地描述角速率、姿态、加速度、速度、位移等需要 3 个坐标系。坐标系 β 相对于坐标系 α 的变化量 $\boldsymbol{x}$ 在坐标系 γ 的投影表示为$\boldsymbol{x}_{\alpha\beta}^{\gamma}$。例如，地球自转在地理系的坐标为

$$\boldsymbol{\omega}_{\mathrm{ie}}^{\mathrm{t}} = \begin{bmatrix} 0 \\ \omega_e \cos L \\ \omega_e \sin L \end{bmatrix} \tag{3-1}$$

其中ω_e是地球自转角速率，L 是纬度。这是地球系 e 相对于惯性系 i 的转动在地理系 t 的投影。在这种表示方法下，一些简单的计算规则如下：

同一个坐标系内表示的变量符合向量加法规则，即

$$\boldsymbol{x}_{AB}^{\gamma} + \boldsymbol{x}_{BC}^{\gamma} = \boldsymbol{x}_{AC}^{\gamma} \tag{3-2}$$

同一个变量在不同坐标系的换算可以用矩阵表示。

$$\boldsymbol{x}_{\alpha\beta}^{\mu} = \boldsymbol{C}_{\gamma}^{\mu}\boldsymbol{x}_{\alpha\beta}^{\gamma} \tag{3-3}$$

坐标变换矩阵表示旋转关系。例如二维的坐标变换矩阵为

$$\boldsymbol{C}_{\mathrm{n}}^{\mathrm{b}} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{3-4}$$

三维的坐标旋转有 3 个自由度，可以看作是类似形式矩阵相乘。

坐标变换矩阵是正交矩阵，逆矩阵是原矩阵的转置

$$\boldsymbol{C}_{\mu}^{\gamma} = \left(\boldsymbol{C}_{\gamma}^{\mu}\right)^{-1} = \left(\boldsymbol{C}_{\gamma}^{\mu}\right)^{\mathrm{T}} \tag{3-5}$$

3.2. 惯性导航

3.2.1. 基本原理

惯性导航的基本原理是：陀螺仪测量角速度，角速度积分得到姿态。加速度计测量加速度，加速度积分得到速度，速度积分得到位置。

实际情况中有一些因素导致上述计算变得复杂。1.需要进行一些坐标系变换。2.需要考虑地球的自转、重力、以及球形形状。

3.2.2. 姿态更新

三维空间有 3 个旋转自由度。类似式(3-4)，依次绕三个坐标轴旋转，则坐标变换矩阵为

$$\boldsymbol{C}_{\mathrm{n}}^{\mathrm{b}} = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3-6}$$

把坐标变换矩阵表示为绕坐标轴分别旋转三次，三次旋转的角度即为欧拉角。旋转的顺序并不是唯一的，也可以定义旋转顺序不同的欧拉角。同一个坐标变换矩阵，在不同的旋转顺序定义下，有不同的欧拉角角度；同样的旋转角度，按照不同的坐标轴顺序旋转，会得到不同的坐标变换矩阵；这个性质称为姿态角的不可交换性。所以使用欧拉角描述姿态时必须规定清楚旋转顺序。本书中欧拉角定义为：初始状态右前上（*xyz*）三轴位于东北天方向，依次绕上轴旋转偏航角，绕右轴旋转俯仰角，绕前轴旋转横滚角。

如果每次旋转的角度很小，则坐标变换矩阵近似为

$$\mathrm{d}C = \begin{bmatrix} 1 & 0 & -\mathrm{d}\theta_y \\ 0 & 1 & 0 \\ \mathrm{d}\theta_y & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \mathrm{d}\theta_x \\ 0 & -\mathrm{d}\theta_x & 1 \end{bmatrix} \begin{bmatrix} 1 & \mathrm{d}\theta_z & 0 \\ -\mathrm{d}\theta_z & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3-7}$$

略去二阶小量，则有

$$\mathrm{d}C=\begin{bmatrix}1 & \mathrm{d}\theta_z & -\mathrm{d}\theta_y\\ -\mathrm{d}\theta_z & 1 & \mathrm{d}\theta_x\\ \mathrm{d}\theta_y & -\mathrm{d}\theta_x & 1\end{bmatrix} \tag{3-8}$$

上式表示了坐标旋力矩阵与旋转角度的关系。如果旋转角度很小，则不必考虑旋转顺序。为了表示的方便，引入角增量反对称矩阵

$$[\boldsymbol{\theta}]=\begin{bmatrix}0 & -\theta_z & \theta_y\\ \theta_z & 0 & -\theta_x\\ -\theta_y & \theta_x & 0\end{bmatrix} \tag{3-9}$$

那么姿态矩阵更新公式为

$$\boldsymbol{C}_{\mathrm{b}}^{\mathrm{i}}(t+T)=\boldsymbol{C}_{\mathrm{b}}^{\mathrm{i}}(t)\lim_{k\to+\infty}\left(\boldsymbol{I}+\frac{[\boldsymbol{\theta}_{\mathrm{ib}}^{\mathrm{b}}]}{k}\right)^k=\boldsymbol{C}_{\mathrm{b}}^{\mathrm{i}}(t)\exp([\boldsymbol{\theta}_{\mathrm{ib}}^{\mathrm{b}}]) \tag{3-10}$$

其中 exp 表示自然常数 e 为底数的指数函数。$\boldsymbol{C}_{\mathrm{b}}^{\mathrm{i}}(t)$是上一时刻的姿态矩阵，$\boldsymbol{C}_{\mathrm{b}}^{\mathrm{i}}(t+T)$是下一时刻的姿态矩阵。上式即姿态更新公式。

利用麦克劳林公式，能得到更便于计算的如下公式

$$\exp([\boldsymbol{\theta}])=\boldsymbol{I}+\frac{\sin|\theta|}{|\theta|}[\boldsymbol{\theta}]+\frac{1-\cos|\theta|}{|\theta|^2}[\boldsymbol{\theta}]^2 \tag{3-11}$$

如果旋转角度较小，同时为了避免分母为 0，可以采用如下近似公式

$$\exp([\boldsymbol{\theta}])\approx\boldsymbol{I}+[\boldsymbol{\theta}] \tag{3-12}$$

根据上述若干公式，使用陀螺仪数据计算得到姿态。

实际导航系统中，为了防止计算误差导致姿态矩阵失去正交性，也为了减少计算量，往往采用四元数代替姿态矩阵进行姿态更新。四元数定义为

$$\boldsymbol{q}=\begin{bmatrix}\cos\frac{\theta}{2} & u_x\sin\frac{\theta}{2} & u_y\sin\frac{\theta}{2} & u_z\sin\frac{\theta}{2}\end{bmatrix}^{\mathrm{T}} \tag{3-13}$$

其中θ是旋转的角度，$[u_x\quad u_y\quad u_z]^{\mathrm{T}}$是旋转轴的单位向量。

四元数也可以表示为

$$\boldsymbol{q}=\cos\frac{\theta}{2}+\boldsymbol{A}\sin\frac{\theta}{2} \tag{3-14}$$

其中 $\boldsymbol{A}$ 是旋转轴的单位向量。

四元数姿态微分方程为

$$\dot{\boldsymbol{q}}=\frac{1}{2}\begin{bmatrix}0 & -\omega_x & -\omega_y & -\omega_z\\ \omega_x & 0 & \omega_z & -\omega_y\\ \omega_y & -\omega_z & 0 & \omega_x\\ \omega_z & \omega_y & -\omega_x & 0\end{bmatrix}\boldsymbol{q} \tag{3-15}$$

引入 4 维的角增量矩阵

$$[\boldsymbol{\theta}]=\begin{bmatrix}0 & -\theta_x & -\theta_y & -\theta_z\\ \theta_x & 0 & \theta_z & -\theta_y\\ \theta_y & -\theta_z & 0 & \theta_x\\ \theta_z & \theta_y & -\theta_x & 0\end{bmatrix} \tag{3-16}$$

四元数更新姿态的公式为

$$\boldsymbol{q}(t+T)=\left(\cos\frac{|\theta|}{2}\boldsymbol{I}+\frac{\sin\frac{|\theta|}{2}}{|\theta|}[\boldsymbol{\theta}]\right)\boldsymbol{q}(t) \tag{3-17}$$

姿态可以用 3*3 矩阵或者 4*1 的四元数表示。本代码包采用四元数计算姿态，这是主流

方法。但是矩阵对于坐标系变换的计算比较方便，所以坐标变换的地方使用了矩阵表示姿态。根据角度增量更新姿态四元数，为 qupdate 函数。

实际导航中计算姿态时，要扣除地球自转的影响。此外，由于地球是球形的，位置变化时地面会向下弯曲，相对于地面的姿态就变了，所以速度会导致一个额外的角速度。

惯性导航需要依次计算姿态、速度、位置。陀螺仪直接测到 b 系相对于 i 系的转动，而导航解算中需要计算 b 系相对于 n 系的姿态。导航系 n 系相对于惯性系 i 系的相对转动包括两个部分：一是地球自转角速率；二是因为地球表面是曲面，载体位置变化会导致 n 系相对 e 系的姿态发生变化，角速率$\boldsymbol{\omega}_{\mathrm{en}}^{\mathrm{n}}$与速度有关

$$\boldsymbol{\omega}_{\mathrm{en}}^{\mathrm{n}}=\begin{bmatrix}-\dfrac{v_{\mathrm{N}}}{R_{\mathrm{m}}+H}\\ \dfrac{v_{\mathrm{E}}}{R_{\mathrm{p}}+H}\\ \dfrac{v_{\mathrm{E}}\tan L}{R_{\mathrm{p}}+H}\end{bmatrix} \tag{3-18}$$

其中R_{m}和R_{p}分别是子午圈和卯酉圈的半径，L 是纬度。

因为

$$\boldsymbol{\omega}_{\mathrm{nb}}^{\mathrm{b}}=\boldsymbol{\omega}_{\mathrm{ib}}^{\mathrm{b}}-\boldsymbol{\omega}_{\mathrm{ie}}^{\mathrm{b}}-\boldsymbol{\omega}_{\mathrm{en}}^{\mathrm{b}} \tag{3-19}$$

所以

$$\boldsymbol{\omega}_{\mathrm{nb}}^{\mathrm{b}}=\boldsymbol{\omega}_{\mathrm{ib}}^{\mathrm{b}}-\boldsymbol{C}_{\mathrm{n}}^{\mathrm{b}}(\boldsymbol{\omega}_{\mathrm{ie}}^{\mathrm{n}}+\boldsymbol{\omega}_{\mathrm{en}}^{\mathrm{n}}) \tag{3-20}$$

根据上述角速率计算$\boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}}$，即姿态。

3.2.3. 速度更新和位置更新

加速度要从传感器的坐标系换算到东北天。本来速度是加速度积分。但是由于地球自转，需要扣除离心力和科里奥利力。此外，加速度计不能区分重力和一般的加速度，所以还需要扣除重力。

加速度为

$$\dot{\boldsymbol{V}}_{\mathrm{en}}^{\mathrm{n}}=\boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}}\boldsymbol{f}_{\mathrm{b}}-2\boldsymbol{\omega}_{\mathrm{ie}}^{\mathrm{n}}\times\boldsymbol{V}_{\mathrm{en}}^{\mathrm{n}}-\boldsymbol{\omega}_{\mathrm{en}}^{\mathrm{n}}\times\boldsymbol{V}_{\mathrm{en}}^{\mathrm{n}}+\boldsymbol{g} \tag{3-21}$$

其中$\boldsymbol{f}_{\mathrm{b}}$是加速度计的数值。根据等效原理，加速度计不能把重力与真正的加速度相区分，所以要扣除地球重力$\boldsymbol{g}$的影响。因为地球是圆的，所以要补偿离心加速度项$\boldsymbol{\omega}_{\mathrm{en}}^{\mathrm{n}}\times\boldsymbol{V}_{\mathrm{en}}^{\mathrm{n}}$。因为地球在自转，所以要补偿科氏加速度$2\boldsymbol{\omega}_{\mathrm{ie}}^{\mathrm{n}}\times\boldsymbol{V}_{\mathrm{en}}^{\mathrm{n}}$。

加速度积分计算得到速度。速度积分计算得到位置。如果用经纬高表示位置，则有

$$\dot{L}=V_{\mathrm{N}}/R_{\mathrm{m}} \tag{3-22}$$

$$\dot{\lambda}=\frac{V_{\mathrm{E}}}{R_{\mathrm{p}}\cos L} \tag{3-23}$$

其中 L 和 λ 分别是纬度和经度，R_{m}和R_{p}分别是当前位置的子午圈和卯酉圈半径。

地球是个椭球，函数 earthmodelupdate 计算两个方向的半径和重力。

3.3. 组合导航

3.3.1. 原理概述

连续计算惯性导航；当获取卫星数据时，采用扩展卡尔曼滤波修正导航误差。

卡尔曼滤波可以理解为：根据方差求权重，做加权平均。

原始的卡尔曼滤波适用于线性系统。因为导航系统不是线性的，所以采用扩展卡尔曼滤波。扩展卡尔曼滤波的主要方法是，选用误差量，利用一阶微分近似为线性系统。滤波得到误差量估计值后，立刻补偿误差。

有的文献把 EKF 算法进一步细化为 ESKF 算法，严格意义上本代码包的方法属于 ESKF 算法。但是大量的文献没有把 EKF 算法进行如此细致的划分，本代码包的算法完全可以说

就是 EKF 算法。

3.3.2. 卡尔曼滤波

比较复杂的系统中，一方面系统具有多个自由度，另一方面被测量随着时间而变化。因此用状态空间方程的形式描述系统的关系，并把加权平均数计算方法用矩阵表示，则得到卡尔曼滤波。

系统表示为：

$$\boldsymbol{x}_k = \boldsymbol{\Phi}\boldsymbol{x}_{k-1} + \boldsymbol{w}_{k-1} \tag{3-24}$$

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k \tag{3-25}$$

其中是$\boldsymbol{x}$状态量，是希望获得而又难以准确测量的量。式(3-24)描述了被测量的变化关系，这里是离散形式。$\boldsymbol{z}$表示量测量，是能测量得到但是包含随机误差的量。式(3-25)描述了量测量与状态量的关系。$\boldsymbol{w}$和$\boldsymbol{v}$是随机噪声。有的系统中$\boldsymbol{w}$和$\boldsymbol{v}$会乘以系数矩阵，但是大多数惯性导航装置的三轴传感器精度大体相当，因此没必要引入标准卡尔曼滤波的$\boldsymbol{\Gamma}$矩阵。

状态量的变化也可以描述为连续方程

$$\dot{\boldsymbol{x}}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} \tag{3-26}$$

如果采样间隔足够小，离散方程与连续方程的关系为

$$\boldsymbol{\Phi} = \boldsymbol{I} + \boldsymbol{F}T \tag{3-27}$$

其中 T 为采样间隔，$\boldsymbol{I}$为单位矩阵。

卡尔曼滤波的解算过程就是根据$\boldsymbol{z}$估计$\boldsymbol{x}$，具体方法如下：

如果不考虑误差，前后时刻的$\boldsymbol{x}$具有关系

$$\widehat{\boldsymbol{X}}_{k|k-1} = \boldsymbol{\Phi}\widehat{\boldsymbol{X}}_{k-1} \tag{3-28}$$

$\widehat{\boldsymbol{X}}_{k-1}$是前一时刻$\boldsymbol{x}$的估计值，$\widehat{\boldsymbol{X}}_{k|k-1}$是推算的后一时刻的$\boldsymbol{x}$。但是因为误差的存在，这个推算并不准确，需要根据$\boldsymbol{z}$修正，因此取

$$\widehat{\boldsymbol{X}}_k = \widehat{\boldsymbol{X}}_{k|k-1} + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\widehat{\boldsymbol{X}}_{k|k-1}) \tag{3-29}$$

其中$\boldsymbol{K}_k$是反映权重的滤波增益。这个增益由如下方法计算

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{\Phi}\boldsymbol{P}_{k-1}\boldsymbol{\Phi}^{\mathrm{T}} + \boldsymbol{Q} \tag{3-30}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^{\mathrm{T}}\left(\boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^{\mathrm{T}} + \boldsymbol{R}\right)^{-1} \tag{3-31}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{P}_{k|k-1}(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})^{\mathrm{T}} + \boldsymbol{K}_k\boldsymbol{R}\boldsymbol{K}_k^{\mathrm{T}} \tag{3-32}$$

其中$\boldsymbol{P}$、$\boldsymbol{Q}$、$\boldsymbol{R}$分别是$\widehat{\boldsymbol{X}}$、$\boldsymbol{w}$、$\boldsymbol{v}$的方差矩阵。

上述公式给出了线性系统的卡尔曼滤波方法。非线性系统可以局部微分而近似为线性系统，采用扩展卡尔曼滤波方法解算。扩展卡曼滤波中的 $\boldsymbol{x}$ 是误差量，扩展卡尔曼滤波获得误差量后，及时修正，使得误差量总维持在较小范围内；在误差量较小时，局部微分得到的线性系统与原始的非线性系统基本一致，卡尔曼滤波能取得较好效果。

代码包采用闭环反馈校正的方式，滤波后修正惯导误差，所以标准卡尔曼滤波中的$\widehat{\boldsymbol{X}}_{k-1}$取 0，简化后的计算公式为

$$\widehat{\boldsymbol{X}}_k = \boldsymbol{K}_k\boldsymbol{z}_k \tag{3-33}$$

导航系统是非线性系统。取扩展卡尔曼滤波的状态量$\boldsymbol{x}$为 15 维向量，包含位置误差、速度误差、姿态误差、陀螺仪零偏、加速度计零偏各 3 各自由度，即

$$\boldsymbol{x} = [\delta L \quad \delta\lambda \quad \delta h \quad \delta v_{\mathrm{E}} \quad \delta v_{\mathrm{N}} \quad \delta v_{\mathrm{U}} \quad \delta\phi_{\mathrm{E}} \quad \delta\phi_{\mathrm{N}} \quad \delta\phi_{\mathrm{U}} \quad \varepsilon_x \quad \varepsilon_y \quad \varepsilon_z \quad \nabla_x \quad \nabla_y \quad \nabla_z]^{\mathrm{T}} \tag{3-34}$$

用扩展卡尔曼滤波进行组合导航的步骤是：1.进行惯性导航解算。2.卫星修正时，比较惯性导航与卫星导航的结果偏差，即$\boldsymbol{z}$。3.用卡尔曼滤波计算$\boldsymbol{x}$。4.根据$\boldsymbol{x}$修正惯性导航的结果，并返回步骤 1。

3.3.3. 组合导航的状态矩阵

惯性和卫星组合导航系统关键在于具体列出状态矩阵$\boldsymbol{\Phi}$，即可实现组合导航的计算。扩

展卡尔曼滤波的矩阵 $\boldsymbol{F}$ 是雅可比矩阵，即偏微分矩阵。根据惯性导航的计算公式，可以得到 $\boldsymbol{F}$ 如下。

$$\boldsymbol{F}=\begin{bmatrix}\boldsymbol{F}_{\mathrm{pp}} & \boldsymbol{F}_{\mathrm{vp}} & \boldsymbol{O}_3 & \boldsymbol{O}_3 & \boldsymbol{O}_3\\ \boldsymbol{F}_{\mathrm{pv}} & \boldsymbol{F}_{\mathrm{vv}} & \boldsymbol{F}_{\mathrm{av}} & \boldsymbol{O}_3 & \boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}}\\ \boldsymbol{F}_{\mathrm{pa}} & \boldsymbol{F}_{\mathrm{va}} & \boldsymbol{F}_{\mathrm{aa}} & -\boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}} & \boldsymbol{O}_3\\ \boldsymbol{O}_3 & \boldsymbol{O}_3 & \boldsymbol{O}_3 & \boldsymbol{O}_3 & \boldsymbol{O}_3\\ \boldsymbol{O}_3 & \boldsymbol{O}_3 & \boldsymbol{O}_3 & \boldsymbol{O}_3 & \boldsymbol{O}_3\end{bmatrix} \tag{3-35}$$

其中每个子矩阵都是 3 阶方阵，$\boldsymbol{O}_3$表示 0 矩阵。

反映位置误差对位置误差影响的子矩阵为

$$\boldsymbol{F}_{\mathrm{pp}}=\begin{bmatrix}0 & 0 & -\dfrac{v_{\mathrm{N}}}{(R_{\mathrm{m}}+h)^2}\\ \dfrac{v_{\mathrm{E}}\sec L\tan L}{R_{\mathrm{p}}+h} & 0 & -\dfrac{v_{\mathrm{E}}\sec L}{\left(R_{\mathrm{p}}+h\right)^2}\\ 0 & 0 & 0\end{bmatrix} \tag{3-36}$$

反映速度误差对位置误差影响的子矩阵为

$$\boldsymbol{F}_{\mathrm{vp}}=\begin{bmatrix}0 & \dfrac{1}{R_{\mathrm{m}}+h} & 0\\ \dfrac{\sec L}{R_{\mathrm{p}}+h} & 0 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{3-37}$$

反映位置误差对速度误差影响的子矩阵为

$$\boldsymbol{F}_{\mathrm{pv}}=\begin{bmatrix}2\omega_{\mathrm{e}}v_{\mathrm{N}}\cos L+2\omega_{\mathrm{e}}v_{\mathrm{U}}\sin L+\dfrac{v_{\mathrm{N}}v_{\mathrm{E}}\sec^2 L}{R_{\mathrm{p}}+h} & 0 & \dfrac{v_{\mathrm{U}}v_{\mathrm{E}}-v_{\mathrm{N}}v_{\mathrm{E}}\tan L}{\left(R_{\mathrm{p}}+h\right)^2}\\ -\left(2\omega_{\mathrm{e}}v_{\mathrm{E}}\cos L+\dfrac{v_{\mathrm{E}}^2\sec^2 L}{R_{\mathrm{p}}+h}\right) & 0 & \dfrac{v_{\mathrm{N}}v_{\mathrm{U}}}{(R_{\mathrm{m}}+h)^2}+\dfrac{v_{\mathrm{E}}^2\tan L}{\left(R_{\mathrm{p}}+h\right)^2}\\ -2v_{\mathrm{E}}\omega_{\mathrm{e}}\sin L & 0 & -\dfrac{v_{\mathrm{N}}^2}{(R_{\mathrm{m}}+h)^2}-\dfrac{v_{\mathrm{E}}^2}{\left(R_{\mathrm{p}}+h\right)^2}\end{bmatrix} \tag{3-38}$$

反映速度误差对速度误差影响的子矩阵为

$$\boldsymbol{F}_{\mathrm{vv}}=\begin{bmatrix}\dfrac{v_{\mathrm{N}}\tan L-v_{\mathrm{U}}}{R_{\mathrm{p}}+h} & 2\omega_{\mathrm{e}}\sin L+\dfrac{v_{\mathrm{E}}\tan L}{R_{\mathrm{p}}+h} & -2\omega_{\mathrm{e}}\cos L-\dfrac{v_{\mathrm{E}}}{R_{\mathrm{p}}+h}\\ -2\omega_{\mathrm{e}}\sin L-\dfrac{2v_{\mathrm{E}}\tan L}{R_{\mathrm{p}}+h} & \dfrac{-v_{\mathrm{U}}}{R_{\mathrm{m}}+h} & \dfrac{-v_{\mathrm{N}}}{R_{\mathrm{m}}+h}\\ 2\left(\omega_{\mathrm{e}}\cos L+\dfrac{v_{\mathrm{E}}}{R_{\mathrm{p}}+h}\right) & \dfrac{2v_{\mathrm{N}}}{R_{\mathrm{m}}+h} & 0\end{bmatrix} \tag{3-39}$$

反映姿态误差对速度误差影响的子矩阵为

$$\boldsymbol{F}_{\mathrm{av}}=\begin{bmatrix}0 & -f_{\mathrm{U}} & f_{\mathrm{N}}\\ f_{\mathrm{U}} & 0 & -f_{\mathrm{E}}\\ -f_{\mathrm{N}} & f_{\mathrm{E}} & 0\end{bmatrix} \tag{3-40}$$

其中f_{E}、f_{N}、f_{U}是换算到 n 系的加速度计数值，即不扣除重力的比力信息。

$$\boldsymbol{f}_{\mathrm{n}}=\begin{bmatrix}f_{\mathrm{E}}\\ f_{\mathrm{N}}\\ f_{\mathrm{U}}\end{bmatrix}=\boldsymbol{C}_{\mathrm{b}}^{\mathrm{n}}\boldsymbol{f}_{\mathrm{b}} \tag{3-41}$$

反映位置误差对姿态误差影响的子矩阵为

$$\boldsymbol{F}_{\mathrm{pa}}=\begin{bmatrix} 0 & 0 & \dfrac{v_{\mathrm{N}}}{(R_{\mathrm{m}}+h)^2} \\ -\omega_{\mathrm{e}}\sin L & 0 & \dfrac{-v_{\mathrm{E}}}{\left(R_{\mathrm{p}}+h\right)^2} \\ \omega_{\mathrm{e}}\cos L+\dfrac{v_{\mathrm{E}}\sec^2 L}{R_{\mathrm{p}}+h} & 0 & \dfrac{-v_{\mathrm{E}}\tan L}{\left(R_{\mathrm{p}}+h\right)^2} \end{bmatrix} \tag{3-42}$$

反映速度误差对姿态误差影响的子矩阵为

$$\boldsymbol{F}_{\mathrm{va}}=\begin{bmatrix} 0 & -\dfrac{1}{R_{\mathrm{m}}+h} & 0 \\ \dfrac{1}{R_{\mathrm{p}}+h} & 0 & 0 \\ \dfrac{\tan L}{R_{\mathrm{p}}+h} & 0 & 0 \end{bmatrix} \tag{3-43}$$

反映姿态误差对姿态误差影响的子矩阵为

$$\boldsymbol{F}_{\mathrm{aa}}=\begin{bmatrix} 0 & \omega_{\mathrm{e}}\sin L+\dfrac{v_{\mathrm{E}}\tan L}{R_{\mathrm{p}}+h} & -\omega_{\mathrm{e}}\cos L-\dfrac{v_{\mathrm{E}}}{R_{\mathrm{p}}+h} \\ -\omega_{\mathrm{e}}\sin L-\dfrac{v_{\mathrm{E}}\tan L}{R_{\mathrm{p}}+h} & 0 & -\dfrac{v_{\mathrm{N}}}{R_{\mathrm{m}}+h} \\ \omega_{\mathrm{e}}\cos L+\dfrac{v_{\mathrm{E}}}{R_{\mathrm{p}}+h} & \dfrac{v_{\mathrm{N}}}{R_{\mathrm{m}}+h} & 0 \end{bmatrix} \tag{3-44}$$

导航计算机每次收到惯性数据时，要计算$\boldsymbol{F}$矩阵，并更新$\boldsymbol{\Phi}$矩阵。导航计算机收到卫星数据时再进行卡尔曼滤波解算，并根据滤波计算得到的误差量修正导航结果。

4. 代码包预览

4.1. 部分代码截图

（由于版本迭代，实际代码可能与截图有轻微差别）

```
void ins_gyroacc(double gx,double gy,double gz,double ax,
{

    MAT gyro(3,1,0);
    MAT acc(3,1,0);

    MAT cbn;
    MAT an;

    double H;

    gyro.num[0][0]=gx;
    gyro.num[1][0]=gy;
    gyro.num[2][0]=gz;
    acc.num[0][0]=ax;
    acc.num[1][0]=ay;
    acc.num[2][0]=az;

    //根据位置更新地球模型。为了减小计算量，以后可以降低这
    earthmodelupdate();

    //准备，补偿传感器误差。注意方向
    acc1=acc+biasacc;
    gyro1=gyro+biasgyro;


    //一、计算姿态
    wien=tWe();
    wenn=tWv();
    cbn=Cbn(qa);
    wnbb=gyro1-(~cbn)*(wien+wenn);//扣除地球自转、扣除速度
    qa=quatupdate(qa,wnbb*dTins);//更新姿态


    //二、计算速度
    accn=cbn*acc1;//更新这个数，以便于卡尔曼滤波的部分使用
    gn.init(3,1,0);
    gn.num[2][0]=(-ge);
    an=accn-((wien+wien+wenn)^tspeed)+gn;
    tspeed=tspeed+dTins*an;//更新速度

    //三、计算位置
    dpos.init(3,1,0);
    H=tpos.num[2][0];
    dpos.num[0][0]=tspeed.num[1][0]/(Rmeri+H);//北向速度得
    dpos.num[1][0]=tspeed.num[0][0]/((Rprim+H)*cos(tpos.n
    dpos.num[2][0]=tspeed.num[2][0];
    tpos=tpos+dTins*dpos;//更新位置
}
```

```
MAT solve(MAT Z,MAT H)
{
    MAT Pkk=Phi*Pk*(~Phi)+Q;
    MAT K=Pkk*(~H)/(H*Pkk*(~H)+R);
    MAT X=K*Z;//因为每次滤波之后补偿了误差，所以状态预测总是0。

    MAT IKH=Eye-K*H;
    Pk=IKH*Pkk*(~IKH)+K*R*(~K);


    Phi.init(15,15,1);


    return X;
}



void sat(double lati,double longi,double height,double ve,double
{
    MAT Z(6,1,0);
    Z.num[0][0]=tpos.num[0][0]-lati;
    Z.num[1][0]=tpos.num[1][0]-longi;
    Z.num[2][0]=tpos.num[2][0]-height;
    Z.num[3][0]=tspeed.num[0][0]-ve;
    Z.num[4][0]=tspeed.num[1][0]-vn;
    Z.num[5][0]=tspeed.num[2][0]-vu;

    MAT H(6,15,1);

    MAT X=solve(Z,H);
    Q.init(15,15,0);


    tpos=tpos-X.submat(0,0,3,1);
    tspeed=tspeed-X.submat(3,0,3,1);
    qa=quatupdate(qa,(~Cbn(qa))*X.submat(6,0,3,1));
    biasgyro=biasgyro-X.submat(9,0,3,1);
    biasacc=biasacc-X.submat(12,0,3,1);
}
```

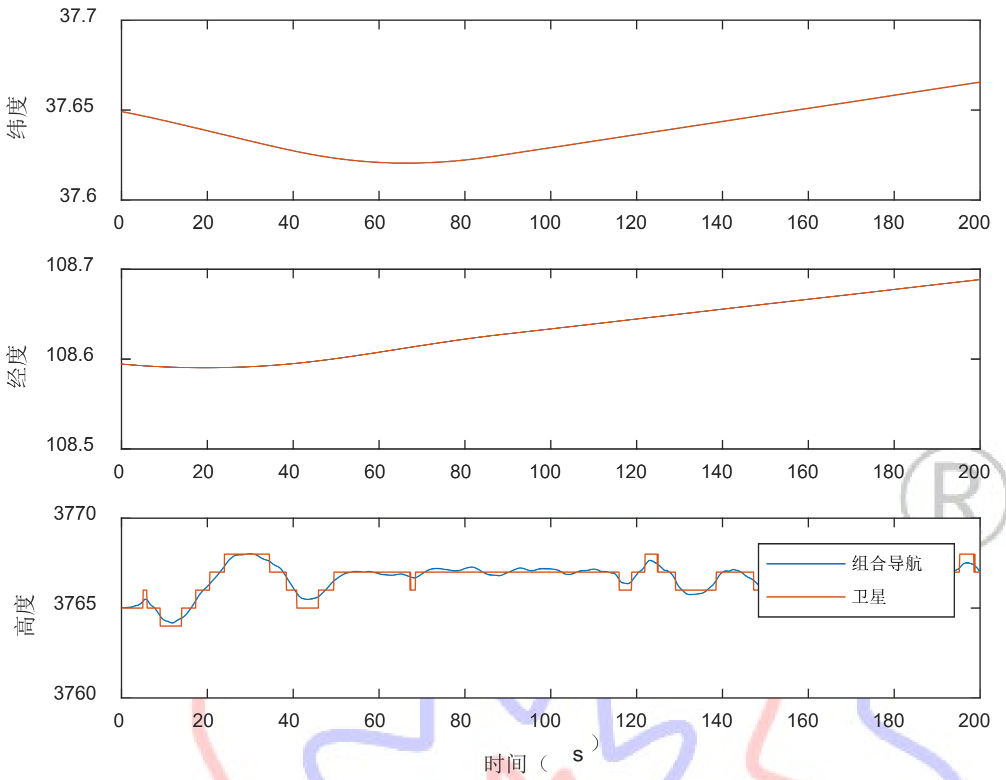
```
//位置对位置影响的子矩阵
Fpp.num[0][2]=RmH1*RmH1*(-vN);//高度对纬度的影响，与北向速度有关
Fpp.num[1][0]=RpH1*vE*secphi*tanphi;//纬度对经度的影响，与纬度有关，
Fpp.num[1][2]=RpH1*RpH1*(-vE)*secphi;//高度对经度的影响，与东向速度有
F.fillsubmat(0,0,Fpp);

//速度对位置影响的子矩阵
Fvp.num[0][1]=RmH1;//北向速度对纬度影响
Fvp.num[1][0]=RpH1*secphi;//东向速度对经度的影响
Fvp.num[2][2]=1;//天向速度对高度的影响
F.fillsubmat(0,3,Fvp);


//位置对速度影响的子矩阵
Fpv.num[0][0]=2*we*cosphi*vN+2*we*sinphi*vU+vN*vE*RpH1*secphi*secphi;
Fpv.num[0][2]=RpH1*RpH1*(vE*vU-vN*vE*tanphi);//高度对东向速度的影响
Fpv.num[1][0]=(-(2*vE*we*cosphi+vE*vE*RpH1*secphi*secphi));//纬度对北
Fpv.num[1][2]=RmH1*RmH1*vN*vU+RpH1*RpH1*vE*vE*tanphi;//高度对北向速度
Fpv.num[2][0]=(-2.0)*vE*we*sinphi;//纬度对天向速度的影响
Fpv.num[2][2]=(-RmH1*RmH1*vN*vN-RpH1*RpH1*vE*vE);//高度对天向速度的影
F.fillsubmat(3,0,Fpv);

//速度对速度影响的子矩阵
Fvv.num[0][0]=(vN*tanphi-vU)*RpH1;//东向速度对东向速度的影响
Fvv.num[0][1]=2.0*we*sinphi+vE*RpH1*tanphi;//北向速度对东向速度的影响
Fvv.num[0][2]=(-2.0)*we*cosphi-vE*RpH1;//天向速度对东向速度的影响
Fvv.num[1][0]=(-2.0)*(we*sinphi+vE*RpH1*tanphi);//东向速度对北向速度
Fvv.num[1][1]=(-RmH1)*vU;//北向速度对北向速度的影响
Fvv.num[1][2]=(-RmH1)*vN;//天向速度对北向速度的影响
Fvv.num[2][0]=2.0*(we*cosphi+vE*RpH1);//东向速度对天向速度的影响//有
Fvv.num[2][1]=2*vN*RmH1;//北向速度对天向速度的影响
F.fillsubmat(3,3,Fvv);

//姿态对速度影响的子矩阵
double fE=accn.num[0][0];
double fN=accn.num[1][0];
double fU=accn.num[2][0];


Fav.num[0][1]=(-fU);//北向角度对东向速度的影响
Fav.num[0][2]=fN;//天向角度对东向速度的影响
Fav.num[1][0]=fU;//东向角度对北向速度的影响
Fav.num[1][2]=(-fE);//天向角度对北向速度的影响
Fav.num[2][0]=(-fN);//东向角度对天向速度的影响
Fav.num[2][1]=fE;//北向角度对天向速度的影响
F.fillsubmat(3,6,Fav);
```

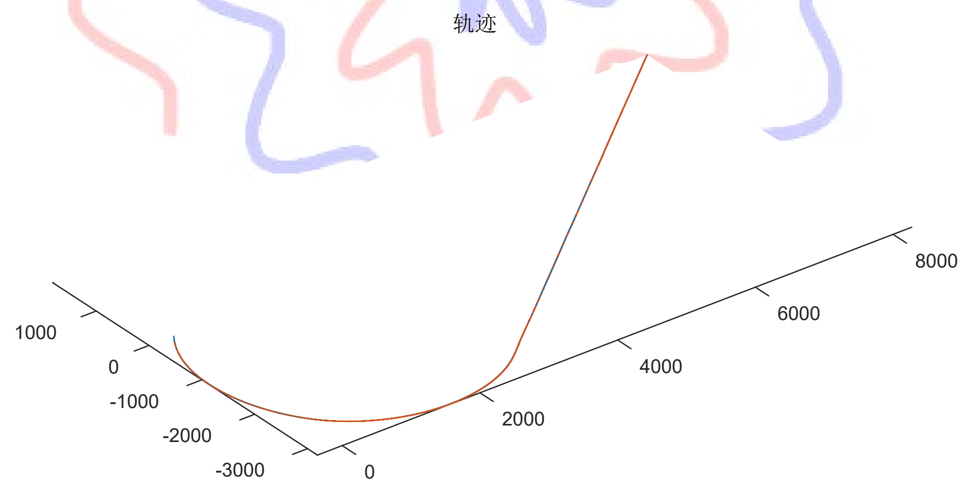
```cpp

class MAT
{
public:
    MAT();
    MAT(int setm,int setn,int kind);//kind=1单位阵，kind=0零矩阵,其它不
    void init(int setm,int setn,int kind);//kind=1单位阵，kind=0零矩阵,


    //这些关键数本应该作为private的。但是为了方便，我也做成了public
    int m;//行数
    int n;//列数
    double num[MAT_MAX][MAT_MAX];//矩阵数据内容


    //特殊的矩阵
    MAT submat(int a,int b,int lm,int ln);//获取矩阵一部分
    void fillsubmat(int a,int b,MAT s);//填充子矩阵



    //向量专用
    double absvec();//这个是向量的长度。不是个别元素的绝对值。
    double square();//向量长度的平方
    friend MAT operator ^(MAT a,MAT b);//叉乘


    //运算
    friend MAT operator *(double k,MAT a);
    friend MAT operator *(MAT a,double k);
    friend MAT operator /(MAT a,double k);
    friend MAT operator *(MAT a,MAT b);
    friend MAT operator +(MAT a,MAT b);
    friend MAT operator -(MAT a,MAT b);
    friend MAT operator ~(MAT a);//转置
    friend MAT operator /(MAT a,MAT b);//a*inv(b)
    friend MAT operator %(MAT a,MAT b);//inv(a)*b


    //MAT inv();//逆矩阵

private:
    //为了用高斯消元法，做的一些函数
    void rowexchange(int a, int b);//交换两行
    void rowmulti(int a,double k);//某一行乘以系数
    void rowadd(int a,int b,double k);//对某一行加减另一行的倍数


    void columnexchange(int a, int b);//交换两列
    void columnmulti(int a,double k);//某一列乘以系数
    void columnadd(int a,int b,double k);//对某一列加减另一列的倍数


};
```

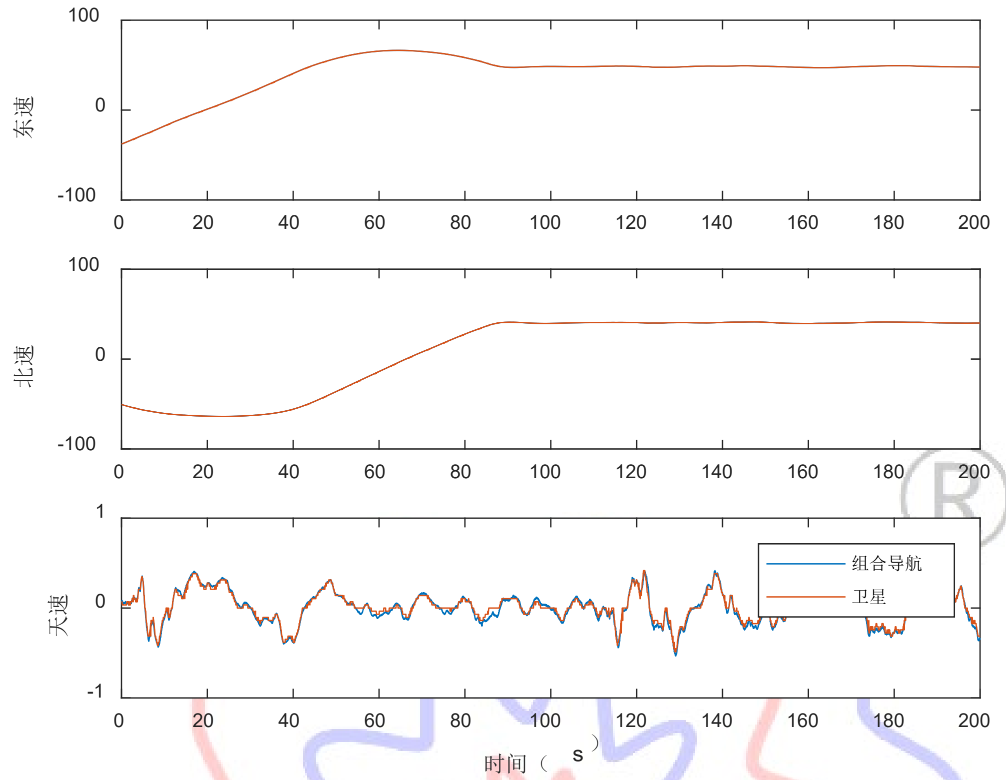
4.2. 输出数据曲线

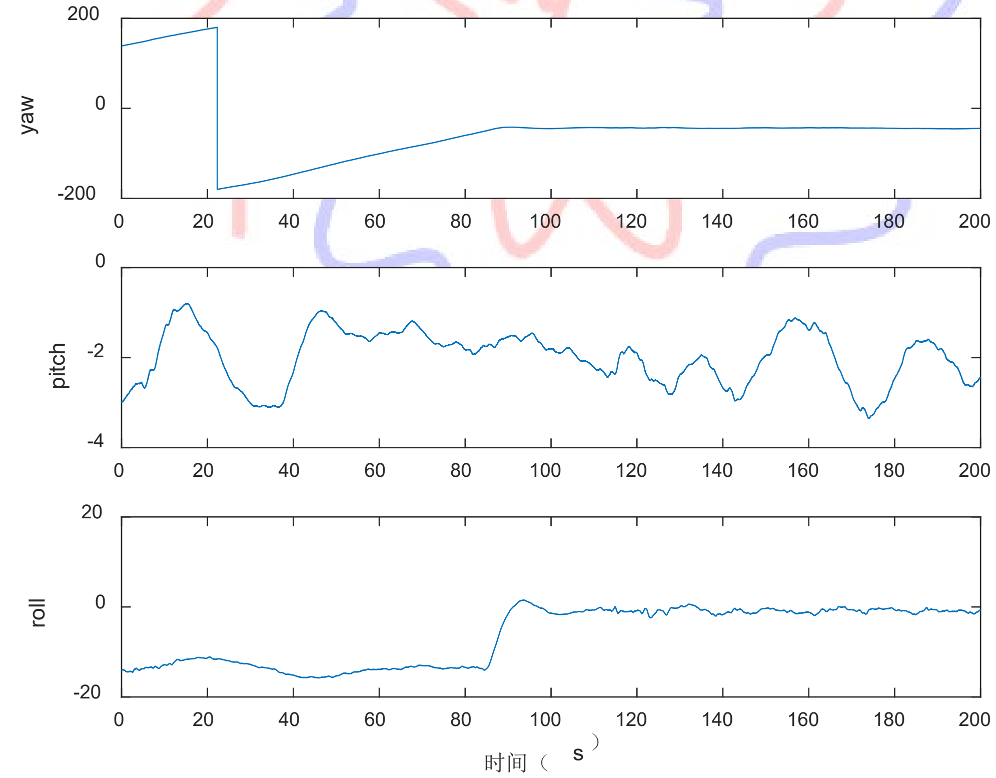
输出数据为 txt 文件，借助 Matlab 绘制曲线。

138.593281,	-2.990332,	-13.988171	-37.823676,	-50.572784,	0.069262	0.657102081,	1.895330379,	3765.000822
138.602132,	-2.989852,	-13.984258	-37.815786,	-50.579683,	0.070872	0.657102049,	1.895330349,	3765.001105
138.611238,	-2.989223,	-13.983118	-37.809041,	-50.585438,	0.070611	0.657102017,	1.895330319,	3765.001388
138.620996,	-2.988699,	-13.984884	-37.799930,	-50.591722,	0.069868	0.657101985,	1.895330289,	3765.001667
138.630975,	-2.989768,	-13.991055	-37.793403,	-50.595704,	0.068866	0.657101953,	1.895330259,	3765.001943
138.641310,	-2.991255,	-13.999024	-37.786767,	-50.602783,	0.067201	0.657101922,	1.895330229,	3765.002212
138.652042,	-2.992262,	-14.006234	-37.779673,	-50.608988,	0.069081	0.657101890,	1.895330199,	3765.002488
138.662192,	-2.992580,	-14.010704	-37.771821,	-50.614341,	0.071019	0.657101858,	1.895330170,	3765.002772
138.672733,	-2.991786,	-14.011916	-37.764718,	-50.620550,	0.072897	0.657101826,	1.895330140,	3765.003064
138.682480,	-2.990583,	-14.011832	-37.757668,	-50.625210,	0.075106	0.657101794,	1.895330110,	3765.003364
138.692300,	-2.989301,	-14.012751	-37.749227,	-50.629480,	0.076032	0.657101762,	1.895330080,	3765.003668
138.701806,	-2.988182,	-14.015506	-37.741183,	-50.636173,	0.073080	0.657101731,	1.895330050,	3765.003960
138.711118,	-2.988187,	-14.017636	-37.735273,	-50.639691,	0.073417	0.657101699,	1.895330020,	3765.004254
138.719963,	-2.989291,	-14.018207	-37.727917,	-50.644368,	0.072139	0.657101667,	1.895329991,	3765.004543
138.728292,	-2.989798,	-14.014371	-37.722870,	-50.650065,	0.071014	0.657101635,	1.895329961,	3765.004827
138.735517,	-2.990094,	-14.007639	-37.714198,	-50.656101,	0.072679	0.657101603,	1.895329931,	3765.005117
138.741718,	-2.989387,	-13.998981	-37.707370,	-50.658980,	0.074149	0.657101571,	1.895329901,	3765.005414
138.747588,	-2.989431,	-13.993644	-37.686202,	-50.678954,	0.072790	0.657101560,	1.895329836,	3765.002725
138.753930,	-2.988426,	-13.987433	-37.679962,	-50.687324,	0.073209	0.657101528,	1.895329806,	3765.003018
138.760895,	-2.987947,	-13.982328	-37.672747,	-50.692184,	0.076493	0.657101496,	1.895329776,	3765.003324
138.767773,	-2.987452,	-13.977163	-37.665986,	-50.697919,	0.076232	0.657101464,	1.895329746,	3765.003629
138.774226,	-2.986334,	-13.971469	-37.657386,	-50.702383,	0.078235	0.657101432,	1.895329717,	3765.003942
138.780410,	-2.984857,	-13.963402	-37.652244,	-50.706730,	0.078518	0.657101400,	1.895329687,	3765.004256


纬度
37.7
37.65
37.6
经度
108.7
108.6
108.5
高度
3770
3765
3760
0
20
40
60
80
100
120
140
160
180
200
组合导航
卫星
时间（s）


轨迹
1000
0
-1000
-2000
-3000
0
2000
4000
6000
8000





5. 常见问题

5.1. 代码容易看懂吗

本代码为了便于学习理解，力求简洁清楚。但是代码包技术含量较高，本店不保证用户能理解代码。用户可以参照前文的代码截图，预先评估自己是否可以理解代码包的内容。

为了防止非法转卖，代码包的注释比较简化。如有难以理解的部分，可以联系本店人工答疑。

本店另有 Matlab 代码包（另行收费）。Matlab 代码比 C++代码更简洁一些，可以供对照学习研究。WMS501、WMS502、WMS503 代码包的计算过程和计算结果相同，只是程序语言不同。

5.2. 代码可以移植到嵌入式吗

代码包主要是纯计算部分，具有跨平台移植的潜力；本代码已经在若干嵌入式项目中应用。但是嵌入式系统调试困难，移植难度较大；根据我店统计，只有约 10%的用户可以自己移植嵌入式成功。建议需要嵌入式程序的用户直接购买我店成品嵌入式软硬件一体商品，而非自己移植。确实需要自行移植嵌入式程序的用户，应该严格注意以下事项：

本店无法远程支援调试嵌入式程序，只能判读离线复算数据。用户应当先将传感器原始数据传输至计算机并记录，用记录的数据进行离线复算。离线复算具有重复性好、便于观察中间变量等优点，容易快速排除大部分问题。在确认离线复算结果正常后，用户再自行调试嵌入式程序，确保嵌入式程序计算过程与离线复算计算过程完全一致。

本代码包采用标准版本组合导航算法，计算量较大。处理器应当具有双精度硬件 FPU；计算性能参照 STM32F7 系列。处理器性能不足的，可以另行购买本店的简易版本组合导航代码包，如 WMS505。

5.3. 组合导航比卫星导航更准吗

组合导航结果基本相当于卫星导航结果的平滑。对于低精度 IMU，组合结果甚至可能不如卫星导航准确。

组合导航与卫星导航相比，主要优势有：1.组合导航的数据刷新率更高。2.组合导航可以输出姿态信息。

5.4. 算法的精度是多少

对于绝大多数情况，组合导航的精度主要取决于传感器精度，算法精度不是导航结果精度的瓶颈。如果用户得到的导航结果精度偏低，应当优先检查传感器精度。

代码包处理附带数据的精度，不能代表代码包处理用户数据的精度。

代码包附带了真实实验数据。真实实验数据的绝对精度难以评估。

如果用户期望在实验中评估绝对精度，有几种可行的方案：1.采用更高精度的导航设备作为位置基准，与自制设备一同实验；比如采用成品高精度组合导航装置，或采用 RTK 卫星导航装置。2.测绘特定轨道位置，沿着特定轨道运动，比如摄像轨道车。

5.5. 怎样判断组合导航的结果是否正确

用户可以尝试故意把代码包的初始姿态改错、然后再运行，以观察组合导航异常时的计算结果，直观体会组合导航结果正常与异常的差别。

5.6. 代码可以处理自己的实验数据吗

本代码包可以处理实验数据，已经有数百个成功案例。但是收集实验数据和处理实验数据都有一定的技术含量。本店不保证用户能成功。

用户收集实验数据时，应注意以下事项：1.IMU 要有足够的采样率，推荐高于 200Hz；采样率过低可能导致完全丧失精度；另外，IMU 数据不能有大量丢失，应该力求均匀完整采集。2.要确认 IMU 的轴向；一些市面销售的传感器其坐标方向与标准定义不符，需要额外换算。3.要记录惯性导航的初始姿态，尤其是使用低精度惯性导航装置时务必记录初始航向。4.IMU 数据与卫星接收机数据要时间对齐；有人分别记录二者数据且无时标，会导致两种数据无法对齐、不能处理；对于通常的运动场景，时间对齐精度最好优于 0.1s。5.要抑制杆臂效应，让 IMU 与卫星天线尽量靠近。

为了方便，建议实验时先原地静止 10 秒左右，再开始运动。

用户处理实验数据时，应注意以下事项：1.数据单位是正确的。2.惯性导航的初值是正确的，这点尤其重要。3.数据中已经剔除了异常数值。

5.7. 坐标系定义不一致怎么办

本代码包采用东北天坐标系。如果有人采用北东地、北天东等，也可以使用本代码包计算，但是需要稍作修改。不同坐标系定义，影响姿态四元数或欧拉角的定义，即重写公式(3-6)，不影响其余计算。采用其他坐标系，只需要换算初始姿态、输出姿态角即可；不要换算传感器轴向等无关量。

5.8. 怎样调卡尔曼滤波参数

根据大量用户的实践结果，结果异常的情况中 80%以上可以归结于其余部分的错误，参数不合适造成的异常只占很少的比例。此外，算法对于卡尔曼滤波参数不敏感，通常情况下即使参数调整 10 到 100 倍，仍然能得到正常的计算结果。所以，当结果显著异常时，应当优先检查其余部分是否正确，而不是调整参数。

Pk 是初始误差的平方。R 是卫星的误差的平方。Pk 和 R 都能直接计算得到，通常不必大调。注意单位换算。例如，初始位置北向误差大约 10 米，则 Pk 的第一个数取$(10/R_e)^2$，其中R_e是地球半径。其余数值计算方法类似。

Q 反映了惯导噪声等误差项的影响，如速率随机游走等。理论上，根据传感器艾伦方差曲线，推算角速率随机游走等误差项的数值，进一步计算得到 Q 的值。但是实际上，因为惯导的误差并没有全部纳入 15 维卡尔曼滤波的方程中，比如不正交误差、g 值敏感系数、温度系数等等，所以不宜直接根据理论推算 Q 值。

实践中，采用人工调参法比较方便。代码包的默认数值，对于很多导航系统是适用的，可以先不调。如果确实需要调整 Q，开始尝试时 Q 尽量小一些，等待滤波器调试稳定后，逐步调大一点点 Q。先确保稳定性，再调整快速性和准确性。

如果惯导的传感器零偏误差较大，可以调大 Pk 和 Q 的最后 6 个数值。如果卫星导航波动较大，则 Q 可以小一点，R 可以大一点。反之亦然。

确有需要的，可以将数据和能运行的代码发送本店，本店人工调参。（可能需要额外收费）

5.9. 卫星导航只有位置没有速度怎么办

此时需要微调代码。卡尔曼滤波参数 R 改为 3*3 矩阵；观测量 Z1 改为 3*1 矩阵；观测方程 H1 改为 3*15 矩阵。如果不修改代码，而是直接取速度为 0，会导致计算结果异常，不要这样做。

6. 著作权和服务

6.1. 工作原理参考什么资料

参考实体书《组合导航应用笔记》，东南大学出版社，2025 年。

讲解视频，哔哩哔哩视频网搜索“大胡子刘师傅”。

6.2. 著作权声明

本店保留著作权。

电路、说明书、全部附属代码（以下简称本代码包）仅限于学习和研究用途的少量使用；包含改编文件、写入嵌入式系统的编译后程序，所有副本总计不得超过 5 份。

本代码包有偿使用。

严禁转卖或公开发布本代码包的全部或一部分。

大规模应用本代码包需要额外取得本店的授权。

对于违反上述要求的用户，本店有权要求停止销售、撤稿、赔偿损失等。

6.3. 服务内容

赠送 30 分钟语音答疑服务，用于解决较为复杂的疑问。

赠送长期文字答疑，用于解决简单的、零散的疑问。

答疑服务仅限直接购买人本人使用。答疑服务不能转让、不能共享。用户需要保留购买凭证截图；丢失购买凭证的，本店可以不提供答疑服务；不是从本店购买的，而是从其他渠道获得代码包的，不提供答疑服务。

本商品技术含量较高，本店不保证能在限时内解答所有疑问。有需要的用户，可以付费购买额外的语音答疑服务。

本店可提供少量的数据判读服务。但是大量的数据判读服务需要额外收费。较为复杂的数据处理，或者定制化修改代码，可能需要额外收费。

上述服务可能需要排队，本店不能保证服务的实时性。

6.4. 联系方式

西安市雁塔区雾膜软件开发站

销售、答疑、定制开发：

微信：（扫码）

雾膜软件

电子邮箱：braun@wmsoft.wang

网站：http://wmsoft.xyz