惯性导航原理
旋转调制实验报告

仪器科学与光电工程学院

13171115　　陈霁阳

同组：

13171109　　杨桐

13171120　　郝嘉麟

13171132　　李绪

13171068　　曹骏

13171066　　朱学伟

2016 年 5 月

目录

一、实验介绍

本次实验采用安装固定在旋转平台上的 IMU（Inertial measuring unit，惯性测量元件）连续采集导航数据。需要注意的是采集数据的过程分为两个阶段：

（1）静态，转台静止；

（2）动态，转台以一设定的角速度绕其转轴转动，我组实验时转动角速度值为 20 度/秒。

两个阶段测量的数据存入同一个文件（5 第六组实验数据.mat），且两个阶段的数据之间是连续的。

最后，在得知初始航向角（75 度）的情况下，利用所得的导航数据解算转台基座（始终静止）的导航参数。

二、解算导航参数步骤

本次实验的数据解算分为两部分，即静态与动态。静态数据的解算方法与第一次实验基本一致，然而动态部分略有不同，由于 IMU 测得的导航数据并不是转台基座的，所以需要进行一次坐标转换，从而又可以分为两种方法：

（1）先由 IMU 测得的数据直接解算出 IMU 以及转台的导航参数，再转换得到转台基座的导航参数；

（2）先将 IMU 测得的导航数据转换到基座，再解算出基座的导航参数。

所以本次数据解算程序有两个：

exp2_1.m

将导航数据分为静态与动态两部分依次解算，动态部分采用直接结算再转换到基座的方法，最后将两部分的结果绘制在一张图中作为静态测量与旋转调制（动态）的比较。

exp2_2.m

只提取出动态部分的数据，分别用对于动态数据的两种解算方法处理，最后将结果绘制在一张图中作为两种方法的比较。

以下做详细介绍。

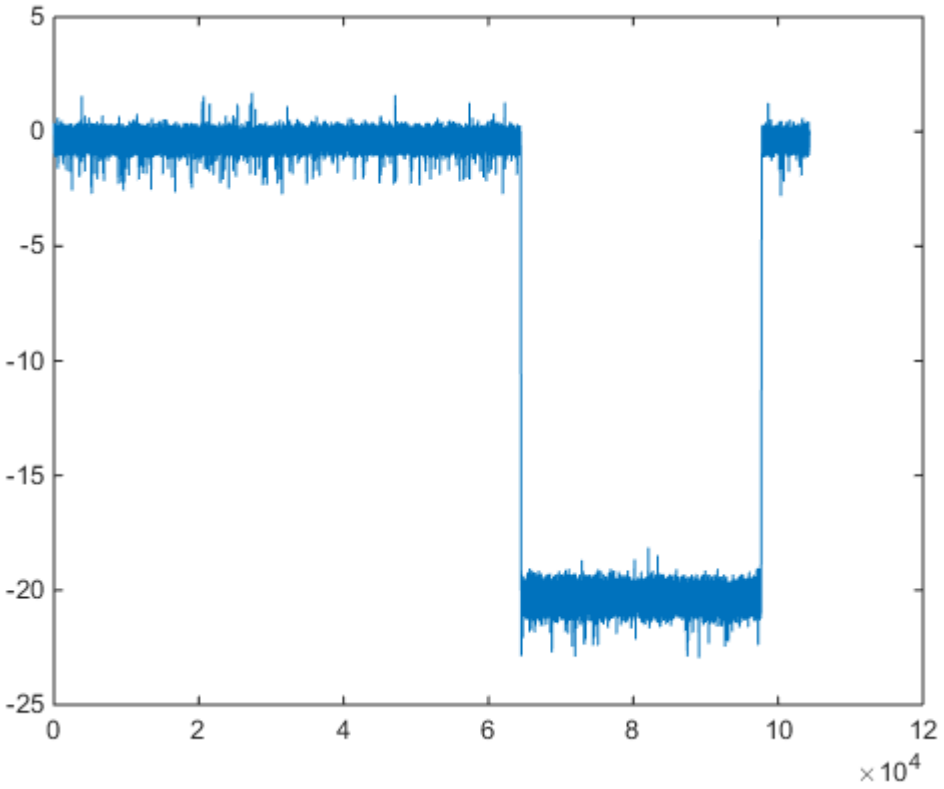
1、原始数据的读取以及预处理

实验数据读入后结构分为 6 列，依次为：

x 轴比力，y 轴比力，z 轴比力，x 轴角速度，y 轴角速度，z 轴角速度

注意比力单位为 g，需要转换为 m/s，角速度单位为度，需要转换为 rad/s。

通过观察 z 轴角速度数据确定静态部分与动态部分的分界点，通过观察，本组数据在 64550 的长度处分界，数据总长度设定为 97500。z 轴角速度测量数据如图：



可以明显得观察到分界位置。

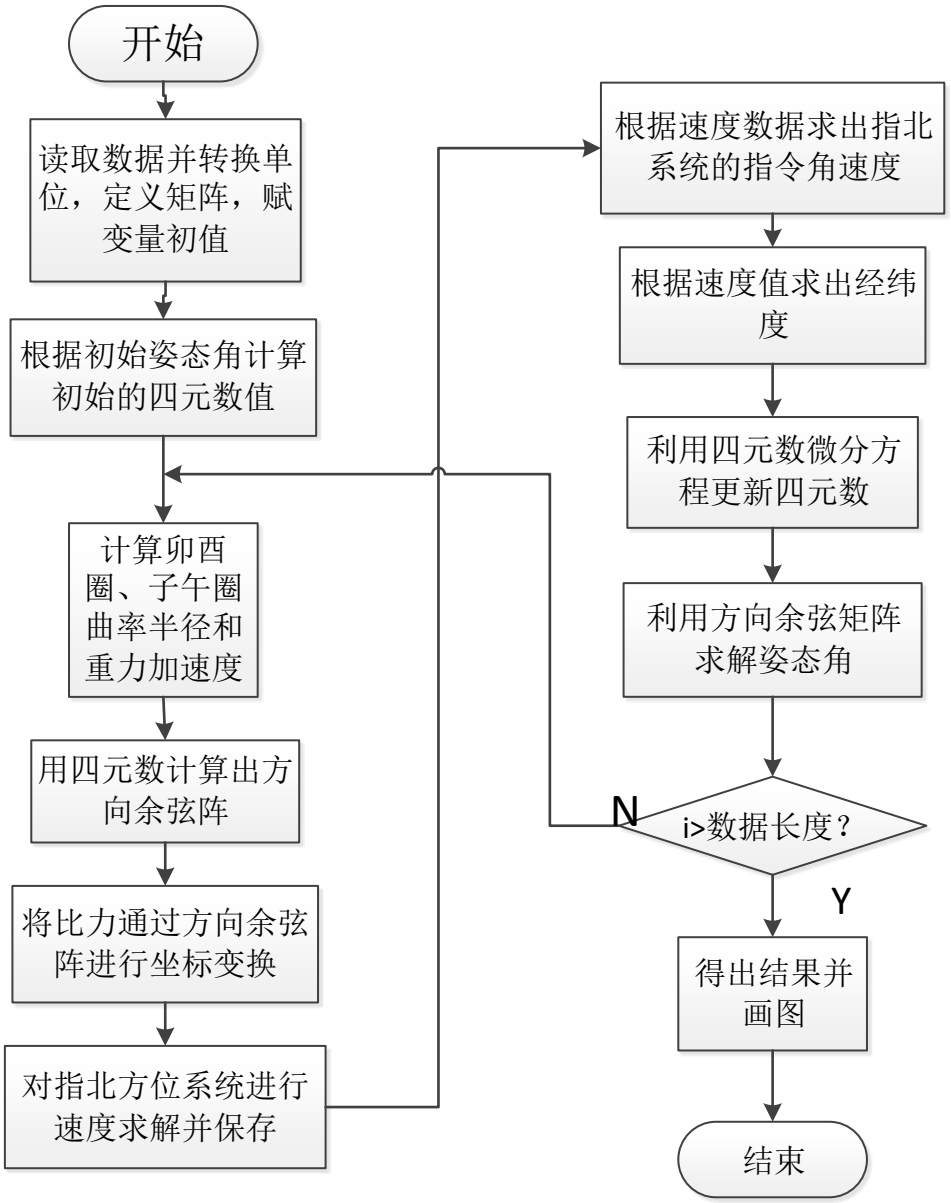
2、初值的设定与计算

初始经纬度以及海拔由 Google Earth 中的查到的实验地点确定，三轴初速度均为 0，偏航角为已知的 75 度，俯仰角与滚转角由**静态部分**测得的 x、y 轴比力的均值计算求得：

俯仰角 $\theta = arcsin(\frac{average(f_y)}{average(f_z)})$

滚转角 $\gamma = arcsin(\frac{average(f_x)}{average(f_z)})$

3、解算导航参数的基本流程



以上流程图为一般的导航数据解算流程，静态数据可以直接套用该流程进行解算，动态（旋转调制）部分数据则需要在解算前或者解算后加入坐标转换的步骤。

流程图中所涉及到的某些具体理论方法如下：

1、根据姿态角计算初始四元数

$$q_0 = \cos\frac{\psi_{ab}}{2}\cos\frac{\theta}{2}\cos\frac{\gamma}{2} - \sin\frac{\psi_{ab}}{2}\sin\frac{\theta}{2}\sin\frac{\gamma}{2}$$
$$q_1 = \cos\frac{\psi_{ab}}{2}\sin\frac{\theta}{2}\cos\frac{\gamma}{2} - \sin\frac{\psi_{ab}}{2}\cos\frac{\theta}{2}\sin\frac{\gamma}{2}$$
$$q_2 = \cos\frac{\psi_{ab}}{2}\cos\frac{\theta}{2}\sin\frac{\gamma}{2} + \sin\frac{\psi_{ab}}{2}\sin\frac{\theta}{2}\cos\frac{\gamma}{2}$$
$$q_3 = \cos\frac{\psi_{ab}}{2}\sin\frac{\theta}{2}\sin\frac{\gamma}{2} + \sin\frac{\psi_{ab}}{2}\cos\frac{\theta}{2}\cos\frac{\gamma}{2}$$

2、利用四元数计算方向余弦矩阵

$$C_t^b = \begin{bmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2+q_0q_3) & 2(q_1q_3-q_0q_2) \\ 2(q_1q_2-q_0q_3) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3+q_0q_1) \\ 2(q_1q_3+q_0q_2) & 2(q_2q_3-q_0q_1) & q_0^2-q_1^2-q_2^2+q_3^2 \end{bmatrix}$$

3、指北方位系统的运动解算：

“平台”指令角速度为：

$$\omega_{it}^t = \begin{bmatrix} -\dfrac{V_y^t}{R_{yt}} \\ \omega_{ie}\cos(L) + \dfrac{V_x^t}{R_{xt}} \\ \omega_{ie}\sin(L) + \dfrac{V_x^t}{R_{xt}}\tan(L) \end{bmatrix}$$

加速度计获得的比力信息 f_{ib}^b 为载体坐标系中各个轴向的比力，而我们需要的比力 f_{it}^t 为地理坐标系中各个轴向的比力，它们之间应用矩阵 C_b^t 做变换：

$$f_{it}^t = C_b^t \times f_{ib}^b$$
$$C_b^t = \left(C_t^b\right)^T = \left(C_t^b\right)^{-1}$$

根据比力信息可以求出各个方向上的加速度：

$$\dot{V}_x^t = f_{ibx}^t + (2\omega_{ie}\sin(L) + \frac{V_x^t}{R_{xt}}\tan(L))V_y^t - (2\omega_{ie}\cos(L) + \frac{V_x^t}{R_{xt}})V_z^t$$
$$\dot{V}_y^t = f_{iby}^t - (2\omega_{ie}\sin(L) + \frac{V_x^t}{R_{xt}}\tan(L))V_x^t + \frac{V_y^t}{R_{yt}}V_z^t$$
$$\dot{V}_z^t = f_{ibz}^t + (2\omega_{ie}\cos(L) + \frac{V_x^t}{R_{xt}})V_x^t + \frac{V_y^t}{R_{yt}}V_y^t - g$$

因此可以求得速度为：

$$V_x^t = \int_0^t \dot{V}_x^t \, dt + V_{x0}^t$$

$$V_y^t = \int_0^t \dot{V}_y^t \, dt + V_{y0}^t$$

载体所在位置的地理纬度 L、经度 λ 可由下列方程求得：

$$L = \int_0^t \frac{V_{yt}}{R_{yt}} dt + L_0$$

$$\lambda = \int_0^t \frac{V_{xt}}{R_{xt}} \sec(L) dt + \lambda_0$$

4、四元数姿态矩阵的更新：

$$\omega_{tb}^b = \omega_{ib}^b - C_t^b \omega_{it}^t$$

式中，ω_{ib}^b 为陀螺所测得的角速度。

用毕卡逼近法更新 C_t^b 的值，T 为采样时间

$$\Delta\theta = T \times \omega_{ib}^b$$

$$[\Delta\theta] = \begin{bmatrix} 0 & -\Delta\theta_x & -\Delta\theta_y & \Delta\theta_z \\ \Delta\theta_x & 0 & \Delta\theta_z & -\Delta\theta_y \\ \Delta\theta_y & -\Delta\theta_z & 0 & \Delta\theta_x \\ \Delta\theta_z & \Delta\theta_y & -\Delta\theta_x & 0 \end{bmatrix}$$

$$\Delta\theta_0^{\,2} = \Delta\theta_x^{\,2} + \Delta\theta_y^{\,2} + \Delta\theta_z^{\,2}$$

$$q(n+1) = \left\{ \left(1 - \frac{(\Delta\theta_0)^2}{8} + \frac{(\Delta\theta_0)^4}{384} \right) I + \left(\frac{1}{2} - \frac{(\Delta\theta_0)^2}{48} \right) [\Delta\theta] \right\} q(n)$$

5、姿态角的求解：

姿态角与姿态矩阵的关系：

$$C_t^b = \begin{bmatrix} \cos\gamma\cos\psi_{ab} - \sin\gamma\sin\theta\sin\psi_{ab} & \cos\gamma\sin\psi_{ab} + \sin\gamma\sin\theta\cos\psi_{ab} & -\sin\gamma\cos\theta \\ -\cos\theta\sin\psi_{ab} & \cos\theta\cos\psi_{ab} & \sin\theta \\ \sin\gamma\cos\psi_{ab} + \cos\gamma\sin\theta\sin\psi_{ab} & -\sin\gamma\sin\psi_{ab} - \cos\gamma\sin\theta\cos\psi_{ab} & \cos\gamma\cos\theta \end{bmatrix}$$

式中 $\theta, \gamma, \psi_{ab}$ 分别为俯仰角，滚转角和偏航角，以逆时针为正方向，而课本上是以顺时针为正，因此需要对课本上的公式进行修改，将 $-\psi_{ab}$ 代入原公式可得现公式。如果记

$$C_t^b = \begin{bmatrix} T_{11} & T_{12} & T_{13} \\ T_{21} & T_{22} & T_{23} \\ T_{31} & T_{32} & T_{33} \end{bmatrix}$$

则由以上两式即可解算出姿态角：

$$\theta = \sin^{-1}(T_{23})$$

$$\gamma = \tan^{-1}\left(-\frac{T_{13}}{T_{33}}\right)$$

$$\varphi = \tan^{-1}\left(\frac{T_{21}}{T_{22}}\right)$$

经过以上流程，就可以解算出静态阶段的导航参数了。

三、动态（旋转调制）导航参数解算

动态部分数据除了多一步坐标转换，其余步骤与静态数据的处理过程基本一致。由于静态部分数据解算进过一段时间后会出现明显误差漂移，所以在对动态（旋转调制）部分数据进行解算的时候需要重新对导航参数赋初值。

1、在 exp2_1.m 中

由于这个程序主要是为了将静态部分解算的结果与旋转调制（动态）的结果进行比较，所以程序中先解算了静态部分数据，然后用相同的方法解算了动态部分数据（得到 IMU 的导航参数），再将之转化为基座的导航参数。

为**先解算再转化。**

因为已知 IMU 相对于基座的旋转角速度为 20 度/s，所以将 IMU 的导航参数转化到基座的导航参数只需要在解算出的 IMU 偏航角上减去转过的角度即可。

另外，在结果图中发现解算出的基座的俯仰角和滚转角呈正弦型，我怀疑是由于 IMU 的安装误差造成的，所以在 IMU 导航参数转化到基座导航参数时，对俯仰角和滚转角也添加了一个与转过角度相关的合适相位与幅度的正弦值进行修正。这部分转化代码为：

```
for i=lengthStill+1:lengthTotalData
    tempAngle = tempAngle + 20 * 1 / 500;
    tempAngle = mod(tempAngle,360);
```

```
    angle = tempAngle * pi / 180;
    base_psi(i) = psi(i) + angle;
    if base_psi(i)>2*pi
        base_psi(i) = base_psi(i) - 2*pi;
    end
    base_theta(1,i) = theta(1,1)+theta(1,i) - theta(1,83000) * sin(angle+75*pi/180);
    base_gamma(1,i) = gamma(1,1)+gamma(1,i) + gamma(1,83000+90/20*500) *
cos(angle+75*pi/180);
end
```

发现修正后结果很好，结果图将在之后的章节分析和展示。

2、在 exp2_2.m 中

本程序是为了比较先直接解算 IMU 导航参数再将结果转化到基座，和现将 IMU 测量数据转化到基座再计算出基座导航参数这两种方法。

先解算再转化的步骤与 exp2_2.m 中一致；

先转化再解算只是在结算前将测得的 z 轴角速度加上设定的 20 度/s，在代码中为：

```
    base_Wibz(1,:)=Wibz(1:lengthTotalData) + 20*pi/180;
```

之后的解算过程与静态部分的解算过程一致。

四、程序代码

1、exp2_1.m

```
% 旋转调制实验
% 采用方法：先解算导航参数，后转化到基座系
clc;clear;
%% 读取数据并储存
data  = load('5 第六组实验数据.mat');
lengthTotalData = 97500;
lengthStill = 64550;
lengthDynamic = lengthTotalData - lengthStill;
fibbx(1,:)=data.temp_data(:,1)*9.80;                          %x 方向的比力数据
fibby(1,:)=data.temp_data(:,2)*9.80;                          %y 方向的比力数据
fibbz(1,:)=data.temp_data(:,3)*9.80;                          %z 方向的比力数据
Wibx(1,:)=data.temp_data(:,4)*pi/180;                  %提取陀螺正东方向角速率并定义
Wiby(1,:)=data.temp_data(:,5)*pi/180;                  %提取陀螺正北方向角速率并定义
```

```
Wibz(1,:)=data.temp_data(:,6)*pi/180;                    %提取陀螺天向角速率并定义

L(1,:)=zeros(1,lengthTotalData);
Lambda(1,:)=zeros(1,lengthTotalData);
Vx(1,:)=zeros(1,lengthTotalData);
Vy(1,:)=zeros(1,lengthTotalData);
Vz(1,:)=zeros(1,lengthTotalData);
Rx(1,:)=zeros(1,lengthTotalData);                        %定义存放卯酉圈曲率半径数据的矩阵
Ry(1,:)=zeros(1,lengthTotalData);                        %定义存放子午圈曲率半径数据的矩阵
psi(1,:)=zeros(1,lengthTotalData);                       %定义存放偏航角数据的矩阵
theta(1,:)=zeros(1,lengthTotalData);                     %定义存放俯仰角数据的矩阵
gamma(1,:)=zeros(1,lengthTotalData);                     %定义存放滚转角数据的矩阵
I=eye(4);                                                %定义四阶矩阵用于计算四元数

%其他相关常量
g0=9.7803267714;
gk1=0.00193185138639;
gk2=0.00669437999013;
Wie=7.292115147e-5;                                      %地球自转角速度
Re=6378245;                                              %长半径
e=1/298.3;                                               %椭圆度
t=1/500;                                                 %采样周期

%定义初始状态
L(1,1)=39.976419/180*pi;                                 %纬度初始值 单位：弧度
Lambda(1,1)=116.340561/180*pi;                           %经度初始值 单位：弧度
H=57;                                                    %高度
Vx(1,1)=0;                                               %初始速度 x 方向分量
Vy(1,1)=0;                                               %初始速度 y 方向分量
Vz(1,1)=0;                                               %初始速度 z 方向分量
psi(1,1)=2*pi-75 /180*pi;                                %偏航角初始值 单位：弧度
theta(1,1)=0;                                            %俯仰角初始值（待计算） 单位：弧度
gamma(1,1)=0;                                            %滚转角初始值（待计算） 单位：弧度

%计算初始值
ave_S_fibbx = mean(fibbx(1,1:60000));
ave_S_fibby = mean(fibby(1,1:60000));
ave_S_fibbz = mean(fibbz(1,1:60000));
theta(1,1) = asin(ave_S_fibby/ave_S_fibbz);              %俯仰角初值（近似求解）
gamma(1,1) = asin(ave_S_fibbx/ave_S_fibbz);              %滚转角初值（近似求解）
%求解四元数系数 q0,q1,q2,q3 的初值
q(1,1)=cos(psi(1,1)/2)*cos(theta(1,1)/2)*cos(gamma(1,1)/2)...
      +sin(psi(1,1)/2)*sin(theta(1,1)/2)*sin(gamma(1,1)/2);      %q0
q(2,1)=cos(psi(1,1)/2)*sin(theta(1,1)/2)*cos(gamma(1,1)/2)...
```

```
    +sin(psi(1,1)/2)*cos(theta(1,1)/2)*sin(gamma(1,1)/2);                %q1
q(3,1)=cos(psi(1,1)/2)*cos(theta(1,1)/2)*sin(gamma(1,1)/2)...
    -sin(psi(1,1)/2)*sin(theta(1,1)/2)*cos(gamma(1,1)/2);                %q2
q(4,1)=cos(psi(1,1)/2)*sin(theta(1,1)/2)*sin(gamma(1,1)/2)...
    -sin(psi(1,1)/2)*cos(theta(1,1)/2)*cos(gamma(1,1)/2);                %q3

%% 循环计算导航参数并更新状态（静止）

for i=1:lengthStill-1
    g=g0*(1+gk1*sin(L(i)^2)*(1-2*H/Re)/sqrt(1-gk2*sin(L(i)^2)));          %计算重力加速度
    Rx(i)=Re/(1-e*(sin(L(i)))^2);                                  %根据纬度计算卯酉圈曲率半径
    Ry(i)=Re/(1+2*e-3*e*(sin(L(i)))^2);                            %根据纬度计算子午圈曲率半径
    %求解四元数姿态矩阵
    q0=q(1,i);q1=q(2,i);q2=q(3,i);q3=q(4,i);
    Ctb=[q0^2+q1^2-q2^2-q3^2, 2*(q1*q2+q0*q3), 2*(q1*q3-q0*q2);
        2*(q1*q2-q0*q3),q2^2-q3^2+q0^2-q1^2,2*(q2*q3+q0*q1);
        2*(q1*q3+q0*q2),2*(q2*q3-q0*q1),q3^2-q2^2-q1^2+q0^2;];
    Cbt=Ctb';
    fibt=Cbt*[fibbx(i);fibby(i);fibbz(i)];                                %比力数据
    fibtx(i)=fibt(1,1);fibty(i)=fibt(2,1);fibtz(i)=fibt(3,1);
    Vx(1,i+1)=(fibtx(i)+(2*Wie*sin(L(i))+Vx(i)*tan(L(i))/Rx(i))*Vy(i)...
            -(2*Wie*cos(L(i))+Vx(i)/Rx(i))*Vz(i))*t+Vx(i);          %计算速度 x 方向分量
    Vy(1,i+1)=(fibty(i)-(2*Wie*sin(L(i))+Vx(i)*tan(L(i))/Rx(i))*Vx(i)...
            +Vy(i)*Vz(i)/Ry(i))*t+Vy(i);                            %计算速度 y 方向分量
    Vz(1,i+1)=(fibtz(i)+(2*Wie*cos(L(i)+Vx(i))/Rx(i))*Vx(i)...
            +Vy(i)*Vy(i)/Ry(i)-g)*t+Vz(i);                          %计算速度 z 方向分量
    Witt=[-Vy(i)/Ry(i);
        Wie*cos(L(i))+Vx(i)/Rx(i);
        Wie*sin(L(i))+Vx(i)*tan(L(i))/Rx(i)];                       %求出平台指令角速度值
    Wibb=[Wibx(i);Wiby(i);Wibz(i)];
    Wtbb=Wibb-Ctb*Witt;                          %将指令角速度转换到平台坐标系,并求解 Wtbb
    L(1,i+1)=t*Vy(i)/Ry(i)+L(i);
    Lambda(1,i+1)=t*Vx(i)/(Rx(i)*cos(L(i)))+ Lambda(i);
    x=Wtbb(1,1)*t;y=Wtbb(2,1)*t;z=Wtbb(3,1)*t;                    %求取迭代矩阵中的各 Δtheta
    A=[0 -x -y -z;x 0 z -y;y -z 0 x;z y -x 0];                    %求取迭代矩阵[Δtheta]
    T=x^2+y^2+z^2;                                                %计算[Δtheta]^2 的
    q(:,i+1)=((1-T/8+T^2/384)*I+(1/2-T/48)*A)*q(:,i);                    %求 q
    theta(i+1)=asin(Ctb(2,3));
    %主值判断
    if(Ctb(2,2)>0)
        if(Ctb(2,1)>=0)
            psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2)));
        else
            psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+2*pi);
```

```
        end
    elseif(Ctb(2,2)<0)
        if(Ctb(2,1)>0)
            psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+pi);
        else
            psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+pi);
        end
    end
    if(Ctb(3,3)>0)
        gamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3));
    elseif(Ctb(1,3)<0)
        gamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3))+pi;
    else
        gamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3))-pi;
    end
end

%% 动态部分初值设定

L(1,lengthStill+1)=39.976419/180*pi;                    %纬度初始值 单位：弧度
Lambda(1,lengthStill+1)=116.340561/180*pi;              %经度初始值 单位：弧度
H=57;                                                   %高度
Vx(1,lengthStill+1)=0;                                  %初始速度 x 方向分量
Vy(1,lengthStill+1)=0;                                  %初始速度 y 方向分量
Vz(1,lengthStill+1)=0;                                  %初始速度 z 方向分量
theta(1,lengthStill+1) = theta(1,1);
gamma(1,lengthStill+1) = gamma(1,1);
psi(1,lengthStill+1)=2*pi-75 /180*pi;                   %偏航角初始值 单位：弧度

%求解四元数系数 q0,q1,q2,q3 的初值
q(1,lengthStill+1)=cos(psi(1,1)/2)*cos(theta(1,1)/2)*cos(gamma(1,1)/2)...
      +sin(psi(1,1)/2)*sin(theta(1,1)/2)*sin(gamma(1,1)/2);          %q0
q(2,lengthStill+1)=cos(psi(1,1)/2)*sin(theta(1,1)/2)*cos(gamma(1,1)/2)...
      +sin(psi(1,1)/2)*cos(theta(1,1)/2)*sin(gamma(1,1)/2);          %q1
q(3,lengthStill+1)=cos(psi(1,1)/2)*cos(theta(1,1)/2)*sin(gamma(1,1)/2)...
      -sin(psi(1,1)/2)*sin(theta(1,1)/2)*cos(gamma(1,1)/2);          %q2
q(4,lengthStill+1)=cos(psi(1,1)/2)*sin(theta(1,1)/2)*sin(gamma(1,1)/2)...
      -sin(psi(1,1)/2)*cos(theta(1,1)/2)*cos(gamma(1,1)/2);          %q3
for i=lengthStill+1:lengthTotalData-1

%% 循环计算导航参数并更新状态（动态）

    g=g0*(1+gk1*sin(L(i)^2)*(1-2*H/Re)/sqrt(1-gk2*sin(L(i)^2)));     %计算重力加速度
    Rx(i)=Re/(1-e*(sin(L(i)))^2);                          %根据纬度计算卯酉圈曲率半径
```

```
Ry(i)=Re/(1+2*e-3*e*(sin(L(i)))^2);                    %根据纬度计算子午圈曲率半径
%求解四元数姿态矩阵
q0=q(1,i);q1=q(2,i);q2=q(3,i);q3=q(4,i);
Ctb=[q0^2+q1^2-q2^2-q3^2, 2*(q1*q2+q0*q3), 2*(q1*q3-q0*q2);
    2*(q1*q2-q0*q3),q2^2-q3^2+q0^2-q1^2,2*(q2*q3+q0*q1);
    2*(q1*q3+q0*q2),2*(q2*q3-q0*q1),q3^2-q2^2-q1^2+q0^2;];
Cbt=Ctb';
fibt=Cbt*[fibbx(i);fibby(i);fibbz(i)];                  %比力数据
fibtx(i)=fibt(1,1);fibty(i)=fibt(2,1);fibtz(i)=fibt(3,1);
Vx(1,i+1)=(fibtx(i)+(2*Wie*sin(L(i))+Vx(i)*tan(L(i))/Rx(i))*Vy(i)...
          -(2*Wie*cos(L(i))+Vx(i)/Rx(i))*Vz(i))*t+Vx(i);    %计算速度 x 方向分量
Vy(1,i+1)=(fibty(i)-(2*Wie*sin(L(i))+Vx(i)*tan(L(i))/Rx(i))*Vx(i)...
          +Vy(i)*Vz(i)/Ry(i))*t+Vy(i);                  %计算速度 y 方向分量
Vz(1,i+1)=(fibtz(i)+(2*Wie*cos(L(i)+Vx(i))/Rx(i))*Vx(i)...
          +Vy(i)*Vy(i)/Ry(i)-g)*t+Vz(i);                %计算速度 z 方向分量
Witt=[-Vy(i)/Ry(i);
    Wie*cos(L(i))+Vx(i)/Rx(i);
    Wie*sin(L(i))+Vx(i)*tan(L(i))/Rx(i)];               %求出平台指令角速度值
Wibb=[Wibx(i);Wiby(i);Wibz(i)];
Wtbb=Wibb-Ctb*Witt;                          %将指令角速度转换到平台坐标系,并求解 Wtbb
L(1,i+1)=t*Vy(i)/Ry(i)+L(i);
Lambda(1,i+1)=t*Vx(i)/(Rx(i)*cos(L(i)))+ Lambda(i);
x=Wtbb(1,1)*t;y=Wtbb(2,1)*t;z=Wtbb(3,1)*t;          %求取迭代矩阵中的各 Δtheta
A=[0 -x -y -z;x 0 z -y;y -z 0 x;z y -x 0];          %求取迭代矩阵[Δtheta]
T=x^2+y^2+z^2;                                      %计算[Δtheta]^2 的
q(:,i+1)=((1-T/8+T^2/384)*I+(1/2-T/48)*A)*q(:,i);   %求 q
theta(i+1)=asin(Ctb(2,3));
%主值判断
if(Ctb(2,2)>0)
    if(Ctb(2,1)>=0)
        psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2)));
    else
        psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+2*pi);
    end
elseif(Ctb(2,2)<0)
    if(Ctb(2,1)>0)
        psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+pi);
    else
        psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+pi);
    end
end
if(Ctb(3,3)>0)
    gamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3));
elseif(Ctb(1,3)<0)
```

```
        gamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3))+pi;
    else
        gamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3))-pi;
    end
end
psi(1,1) = 75 /180*pi;
psi(1,lengthStill+1)=75 /180*pi;

%% 平台系到基座系

base_Vx(1,:)=zeros(1,lengthTotalData);
base_Vy(1,:)=zeros(1,lengthTotalData);
base_Vz(1,:)=zeros(1,lengthTotalData);
base_theta(1,:)=zeros(1,lengthTotalData);                    %定义存放俯仰角数据的矩阵
base_gamma(1,:)=zeros(1,lengthTotalData);                    %定义存放滚转角数据的矩阵
base_psi(1,:)=zeros(1,lengthTotalData);                      %定义存放偏航角数据的矩阵
tempAngle = 0;

for i=lengthStill+1:lengthTotalData
    tempAngle = tempAngle + 20 * 1 / 500;
    tempAngle = mod(tempAngle,360);
    angle = tempAngle * pi / 180;
    base_psi(i) = psi(i) + angle;
    if base_psi(i)>2*pi
        base_psi(i) = base_psi(i) - 2*pi;
    end
    base_theta(1,i) = theta(1,1)+theta(1,i) - theta(1,83000) * sin(angle+75*pi/180);
    base_gamma(1,i) = gamma(1,1)+gamma(1,i) + gamma(1,83000+90/20*500) *
cos(angle+75*pi/180);
end

%% 统一绘图

ts=linspace(0,(lengthStill-1)/500,lengthStill);
td=linspace((lengthStill)/500,(lengthTotalData-1)/500,lengthDynamic);

figure, hold on;
plot(Lambda(1:lengthStill)*180/pi,L(1:lengthStill)*180/pi,'b');
plot(Lambda(lengthStill+1:lengthTotalData)*180/pi,L(lengthStill+1:lengthTotalData)*180/pi,'r
');
title('经纬度位置曲线'); xlabel('经度/° '); ylabel('纬度/° '); legend('静态','动态
','Location','northwest');
grid on; axis equal; hold off;
```

```
figure, hold on;
plot(ts,Vx(1:lengthStill),'b'); plot(td,Vx(lengthStill+1:lengthTotalData),'r');
title('东西方向速度');xlabel('时间/s');ylabel('速度/m/s');legend('静态','动态
','Location','northwest');
grid on; hold off;

figure, hold on;
plot(ts,Vy(1:lengthStill),'b'); plot(td,Vy(lengthStill+1:lengthTotalData),'r');
title('南北方向速度');xlabel('时间/s');ylabel('速度/m/s');legend('静态','动态
','Location','northwest');
grid on; hold off;

figure, hold on;
plot(ts,theta(1:lengthStill)*180/pi,'b');
plot(td,theta(lengthStill+1:lengthTotalData)*180/pi,'g');
plot(td,base_theta(lengthStill+1:lengthTotalData)*180/pi,'r');
title('俯仰角');xlabel('时间/s');ylabel('度');legend('静态','平台','基座
','Location','northeast');
grid on; hold off;

figure, hold on;
plot(ts,gamma(1:lengthStill)*180/pi,'b');
plot(td,gamma(lengthStill+1:lengthTotalData)*180/pi,'g');
plot(td,base_gamma(lengthStill+1:lengthTotalData)*180/pi,'r');
title('滚转角');xlabel('时间/s');ylabel('度');legend('静态','平台','基座
','Location','northwest');
grid on; hold off;

figure, hold on;
plot(ts,psi(1:lengthStill)*180/pi,'b');
plot(td,psi(lengthStill+1:lengthTotalData)*180/pi,'g');
plot(td,base_psi(lengthStill+1:lengthTotalData)*180/pi,'r');
title('偏航角');xlabel('时间/s');ylabel('度');legend('静态','平台','基座
','Location','northwest');
grid on; hold off;
```

2、exp2_2.m

```
% 旋转调制实验
% 采用方法：先转化到基座系，后解算导航参数。只对动态部分进行解算，与之前的结果作比较
clc;clear;
    %% 读取数据并储存
data  = load('5 第六组实验数据.mat');
```

```
lengthTotalData = 97500;
lengthStill = 64550;
lengthDynamic = lengthTotalData - lengthStill;
fibbx(1,:)=data.temp_data(:,1)*9.80;                    %x 方向的比力数据
fibby(1,:)=data.temp_data(:,2)*9.80;                    %y 方向的比力数据
fibbz(1,:)=data.temp_data(:,3)*9.80;                    %z 方向的比力数据
Wibx(1,:)=data.temp_data(:,4)*pi/180;                   %提取陀螺正东方向角速率并定义
Wiby(1,:)=data.temp_data(:,5)*pi/180;                   %提取陀螺正北方向角速率并定义
Wibz(1,:)=data.temp_data(:,6)*pi/180;                   %提取陀螺天向角速率并定义

L(1,:)=zeros(1,lengthTotalData);
Lambda(1,:)=zeros(1,lengthTotalData);
Vx(1,:)=zeros(1,lengthTotalData);
Vy(1,:)=zeros(1,lengthTotalData);
Vz(1,:)=zeros(1,lengthTotalData);
Rx(1,:)=zeros(1,lengthTotalData);                       %定义存放卯酉圈曲率半径数据的矩阵
Ry(1,:)=zeros(1,lengthTotalData);                       %定义存放子午圈曲率半径数据的矩阵
psi(1,:)=zeros(1,lengthTotalData);                      %定义存放偏航角数据的矩阵
theta(1,:)=zeros(1,lengthTotalData);                    %定义存放俯仰角数据的矩阵
gamma(1,:)=zeros(1,lengthTotalData);                    %定义存放滚转角数据的矩阵
I=eye(4);                                               %定义四阶矩阵用于计算四元数

%其他相关常量
g0=9.7803267714;
gk1=0.00193185138639;
gk2=0.00669437999013;
Wie=7.292115147e-5;                                     %地球自转角速度
Re=6378245;                                             %长半径
e=1/298.3;                                              %椭圆度
t=1/500;                                                %采样周期

%定义初始状态
L(1,1)=39.976419/180*pi;                                %纬度初始值 单位：弧度
Lambda(1,1)=116.340561/180*pi;                          %经度初始值 单位：弧度
H=57;                                                   %高度
Vx(1,1)=0;                                              %初始速度 x 方向分量
Vy(1,1)=0;                                              %初始速度 y 方向分量
Vz(1,1)=0;                                              %初始速度 z 方向分量
psi(1,1)=2*pi-75 /180*pi;                               %偏航角初始值 单位：弧度
theta(1,1)=0;                                           %俯仰角初始值（待计算） 单位：弧度
gamma(1,1)=0;                                           %滚转角初始值（待计算） 单位：弧度

    %% 动态部分初值设定
ave_S_fibbx = mean(fibbx(1,1:60000));
```

```
ave_S_fibby = mean(fibby(1,1:60000));

ave_S_fibbz = mean(fibbz(1,1:60000));

theta(1,1) = asin(ave_S_fibby/ave_S_fibbz);                    %俯仰角初值（近似求解）
gamma(1,1) = asin(ave_S_fibbx/ave_S_fibbz);                    %滚转角初值（近似求解）
L(1,lengthStill+1)=39.976419/180*pi;                           %纬度初始值 单位：弧度
Lambda(1,lengthStill+1)=116.340561/180*pi;                     %经度初始值 单位：弧度
H=57;                                                          %高度
Vx(1,lengthStill+1)=0;                                         %初始速度 x 方向分量
Vy(1,lengthStill+1)=0;                                         %初始速度 y 方向分量
Vz(1,lengthStill+1)=0;                                         %初始速度 z 方向分量
theta(1,lengthStill+1) = theta(1,1);
gamma(1,lengthStill+1) = gamma(1,1);
psi(1,lengthStill+1)=2*pi-75 /180*pi;                          %偏航角初始值 单位：弧度

%求解四元数系数 q0,q1,q2,q3 的初值
q(1,lengthStill+1)=cos(psi(1,1)/2)*cos(theta(1,1)/2)*cos(gamma(1,1)/2)...
      +sin(psi(1,1)/2)*sin(theta(1,1)/2)*sin(gamma(1,1)/2);          %q0
q(2,lengthStill+1)=cos(psi(1,1)/2)*sin(theta(1,1)/2)*cos(gamma(1,1)/2)...
      +sin(psi(1,1)/2)*cos(theta(1,1)/2)*sin(gamma(1,1)/2);          %q1
q(3,lengthStill+1)=cos(psi(1,1)/2)*cos(theta(1,1)/2)*sin(gamma(1,1)/2)...
      -sin(psi(1,1)/2)*sin(theta(1,1)/2)*cos(gamma(1,1)/2);          %q2
q(4,lengthStill+1)=cos(psi(1,1)/2)*sin(theta(1,1)/2)*sin(gamma(1,1)/2)...
      -sin(psi(1,1)/2)*cos(theta(1,1)/2)*cos(gamma(1,1)/2);          %q3


    %% 循环计算导航参数并更新状态（先解算再转化）
for i=lengthStill+1:lengthTotalData-1
    g=g0*(1+gk1*sin(L(i)^2)*(1-2*H/Re)/sqrt(1-gk2*sin(L(i)^2)));    %计算重力加速度
    Rx(i)=Re/(1-e*(sin(L(i)))^2);                                  %根据纬度计算卯酉
圈曲率半径
    Ry(i)=Re/(1+2*e-3*e*(sin(L(i)))^2);                            %根据纬度计算子午
圈曲率半径
    %求解四元数姿态矩阵
    q0=q(1,i);q1=q(2,i);q2=q(3,i);q3=q(4,i);
    Ctb=[q0^2+q1^2-q2^2-q3^2, 2*(q1*q2+q0*q3), 2*(q1*q3-q0*q2);
        2*(q1*q2-q0*q3),q2^2-q3^2+q0^2-q1^2,2*(q2*q3+q0*q1);
        2*(q1*q3+q0*q2),2*(q2*q3-q0*q1),q3^2-q2^2-q1^2+q0^2;];
    Cbt=Ctb';
    fibt=Cbt*[fibbx(i);fibby(i);fibbz(i)];                         %比力数据
    fibtx(i)=fibt(1,1);fibty(i)=fibt(2,1);fibtz(i)=fibt(3,1);
    Vx(1,i+1)=(fibtx(i)+(2*Wie*sin(L(i))+Vx(i)*tan(L(i))/Rx(i))*Vy(i)...
            -(2*Wie*cos(L(i))+Vx(i)/Rx(i))*Vz(i))*t+Vx(i);          %计算速度 x 方向分量
    Vy(1,i+1)=(fibty(i)-(2*Wie*sin(L(i))+Vx(i)*tan(L(i))/Rx(i))*Vx(i)...
```

```
            +Vy(i)*Vz(i)/Ry(i))*t+Vy(i);                                %计算速度 y 方向分量
    Vz(1,i+1)=(fibtz(i)+(2*Wie*cos(L(i)+Vx(i))/Rx(i))*Vx(i)...
            +Vy(i)*Vy(i)/Ry(i)-g)*t+Vz(i);                              %计算速度 z 方向分量
    Witt=[-Vy(i)/Ry(i);
        Wie*cos(L(i))+Vx(i)/Rx(i);
        Wie*sin(L(i))+Vx(i)*tan(L(i))/Rx(i)];                           %求出平台指令角速度值
    Wibb=[Wibx(i);Wiby(i);Wibz(i)];
    Wtbb=Wibb-Ctb*Witt;                                %将指令角速度转换到平台坐标系,并求解 Wtbb
    L(1,i+1)=t*Vy(i)/Ry(i)+L(i);
    Lambda(1,i+1)=t*Vx(i)/(Rx(i)*cos(L(i)))+ Lambda(i);
    x=Wtbb(1,1)*t;y=Wtbb(2,1)*t;z=Wtbb(3,1)*t;                      %求取迭代矩阵中的各 Δtheta
    A=[0 -x -y -z;x 0 z -y;y -z 0 x;z y -x 0];                      %求取迭代矩阵[Δtheta]
    T=x^2+y^2+z^2;                                                  %计算[Δtheta]^2 的
    q(:,i+1)=((1-T/8+T^2/384)*I+(1/2-T/48)*A)*q(:,i);               %求 q
    theta(i+1)=asin(Ctb(2,3));
    %主值判断
    if(Ctb(2,2)>0)
        if(Ctb(2,1)>=0)
            psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2)));
        else
            psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+2*pi);
        end
    elseif(Ctb(2,2)<0)
        if(Ctb(2,1)>0)
            psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+pi);
        else
            psi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+pi);
        end
    end
    if(Ctb(3,3)>0)
        gamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3));
    elseif(Ctb(1,3)<0)
        gamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3))+pi;
    else
        gamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3))-pi;
    end
end
psi(1,1) = 75 /180*pi;
psi(1,lengthStill+1)=75 /180*pi;

%平台系到基座系
base_Vx(1,:)=zeros(1,lengthTotalData);
base_Vy(1,:)=zeros(1,lengthTotalData);
base_Vz(1,:)=zeros(1,lengthTotalData);
```

```
base_theta(1,:)=zeros(1,lengthTotalData);                    %定义存放俯仰角数据的矩阵
base_gamma(1,:)=zeros(1,lengthTotalData);                    %定义存放滚转角数据的矩阵
base_psi(1,:)=zeros(1,lengthTotalData);                      %定义存放偏航角数据的矩阵
tempAngle = 0;

for i=lengthStill+1:lengthTotalData
    tempAngle = tempAngle + 20 * 1 / 500;
    tempAngle = mod(tempAngle,360);
    angle = tempAngle * pi / 180;
    base_psi(i) = psi(i) + angle;
    if base_psi(i)>2*pi
        base_psi(i) = base_psi(i) - 2*pi;
    end
    base_theta(1,i) = theta(1,1)+theta(1,i) - theta(1,83000) * sin(angle+75*pi/180);
    base_gamma(1,i) = gamma(1,1)+gamma(1,i) + gamma(1,83000+90/20*500) *
cos(angle+75*pi/180);
end
    %% 先转化再解算
base_fibbx(1,:)=fibbx(1:lengthTotalData);                    %x 方向的比力数据
base_fibby(1,:)=fibby(1:lengthTotalData);                    %y 方向的比力数据
base_fibbz(1,:)=fibbz(1:lengthTotalData);                    %z 方向的比力数据
base_Wibx(1,:)=Wibx(1:lengthTotalData);                  %提取陀螺正东方向角速率并定义
base_Wiby(1,:)=Wiby(1:lengthTotalData);                  %提取陀螺正北方向角速率并定义
base_Wibz(1,:)=Wibz(1:lengthTotalData) + 20*pi/180;
NL(1,:)=zeros(1,lengthTotalData);
NLambda(1,:)=zeros(1,lengthTotalData);
NVx(1,:)=zeros(1,lengthTotalData);
NVy(1,:)=zeros(1,lengthTotalData);
NVz(1,:)=zeros(1,lengthTotalData);
Npsi(1,:)=zeros(1,lengthTotalData);                          %定义存放偏航角数据的矩阵
Ntheta(1,:)=zeros(1,lengthTotalData);                        %定义存放俯仰角数据的矩阵
Ngamma(1,:)=zeros(1,lengthTotalData);                        %定义存放滚转角数据的矩阵
%初值设定
Ntheta(1,lengthStill+1) = theta(1,1);
Ngamma(1,lengthStill+1) = gamma(1,1);
Npsi(1,lengthStill+1)=2*pi-75 /180*pi;                       %偏航角初始值 单位：弧度
NL(1,lengthStill+1)=39.976419/180*pi;                        %纬度初始值 单位：弧度
NLambda(1,lengthStill+1)=116.340561/180*pi;                  %经度初始值 单位：弧度
H=57;
%求解四元数系数 q0,q1,q2,q3 的初值
q(1,lengthStill+1)=cos(Npsi(1,lengthStill+1)/2)*cos(Ntheta(1,lengthStill+1)/2)*cos(Ngamma(1,
lengthStill+1)/2)...
```

```
+sin(Npsi(1,lengthStill+1)/2)*sin(Ntheta(1,lengthStill+1)/2)*sin(Ngamma(1,lengthStill+1)/2);
        %q0
q(2,lengthStill+1)=cos(Npsi(1,lengthStill+1)/2)*sin(Ntheta(1,lengthStill+1)/2)*cos(Ngamma(1,
lengthStill+1)/2)...

+sin(Npsi(1,lengthStill+1)/2)*cos(Ntheta(1,lengthStill+1)/2)*sin(Ngamma(1,lengthStill+1)/2);
        %q1
q(3,lengthStill+1)=cos(Npsi(1,lengthStill+1)/2)*cos(Ntheta(1,lengthStill+1)/2)*sin(Ngamma(1,
lengthStill+1)/2)...
       -
sin(Npsi(1,lengthStill+1)/2)*sin(Ntheta(1,lengthStill+1)/2)*cos(Ngamma(1,lengthStill+1)/2);
       %q2
q(4,lengthStill+1)=cos(Npsi(1,lengthStill+1)/2)*sin(Ntheta(1,lengthStill+1)/2)*sin(Ngamma(1,
lengthStill+1)/2)...
       -
sin(Npsi(1,lengthStill+1)/2)*cos(Ntheta(1,lengthStill+1)/2)*cos(Ngamma(1,lengthStill+1)/2);
       %q3
%循环计算导航参数并更新状态
for i=lengthStill+1:lengthTotalData-1
    g=g0*(1+gk1*sin(NL(i)^2)*(1-2*H/Re)/sqrt(1-gk2*sin(NL(i)^2)));          %计算重力加速度
    Rx(i)=Re/(1-e*(sin(NL(i)))^2);                                  %根据纬度计算卯酉圈曲率半径
    Ry(i)=Re/(1+2*e-3*e*(sin(NL(i)))^2);                            %根据纬度计算子午圈曲率半径
    %求解四元数姿态矩阵
    q0=q(1,i);q1=q(2,i);q2=q(3,i);q3=q(4,i);
    Ctb=[q0^2+q1^2-q2^2-q3^2, 2*(q1*q2+q0*q3), 2*(q1*q3-q0*q2);
        2*(q1*q2-q0*q3),q2^2-q3^2+q0^2-q1^2,2*(q2*q3+q0*q1);
        2*(q1*q3+q0*q2),2*(q2*q3-q0*q1),q3^2-q2^2-q1^2+q0^2;];
    Cbt=Ctb';
    fibt=Cbt*[base_fibbx(i);base_fibby(i);base_fibbz(i)];                   %比力数据
    fibtx(i)=fibt(1,1);fibty(i)=fibt(2,1);fibtz(i)=fibt(3,1);
    NVx(1,i+1)=(fibtx(i)+(2*Wie*sin(NL(i))+NVx(i)*tan(NL(i))/Rx(i))*NVy(i)...
            -(2*Wie*cos(NL(i))+NVx(i)/Rx(i))*NVz(i))*t+NVx(i);          %计算速度 x 方向分量
    NVy(1,i+1)=(fibty(i)-(2*Wie*sin(NL(i))+NVx(i)*tan(NL(i))/Rx(i))*NVx(i)...
            +NVy(i)*NVz(i)/Ry(i))*t+NVy(i);                             %计算速度 y 方向分量
    NVz(1,i+1)=(fibtz(i)+(2*Wie*cos(NL(i)+NVx(i))/Rx(i))*NVx(i)...
            +NVy(i)*NVy(i)/Ry(i)-g)*t+NVz(i);                           %计算速度 z 方向分量
    Witt=[-NVy(i)/Ry(i);
        Wie*cos(NL(i))+NVx(i)/Rx(i);
        Wie*sin(NL(i))+NVx(i)*tan(NL(i))/Rx(i)];                    %求出平台指令角速度值
    Wibb=[base_Wibx(i);base_Wiby(i);base_Wibz(i)];
    Wtbb=Wibb-Ctb*Witt;                             %将指令角速度转换到平台坐标系,并求解 Wtbb
    NL(1,i+1)=t*NVy(i)/Ry(i)+NL(i);
    NLambda(1,i+1)=t*NVx(i)/(Rx(i)*cos(NL(i)))+ NLambda(i);
```

```
    x=Wtbb(1,1)*t;y=Wtbb(2,1)*t;z=Wtbb(3,1)*t;                    %求取迭代矩阵中的各 Δtheta
    A=[0 -x -y -z;x 0 z -y;y -z 0 x;z y -x 0];                    %求取迭代矩阵[Δtheta]
    T=x^2+y^2+z^2;                                               %计算[Δtheta]^2 的
    q(:,i+1)=((1-T/8+T^2/384)*I+(1/2-T/48)*A)*q(:,i);             %求 q
    Ntheta(i+1)=asin(Ctb(2,3));
    %主值判断
    if(Ctb(2,2)>0)
        if(Ctb(2,1)>=0)
            Npsi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2)));
        else
            Npsi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+2*pi);
        end
    elseif(Ctb(2,2)<0)
        if(Ctb(2,1)>0)
            Npsi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+pi);
        else
            Npsi(i+1)=2*pi-(atan(Ctb(2,1)/Ctb(2,2))+pi);
        end
    end
    if(Ctb(3,3)>0)
        Ngamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3));
    elseif(Ctb(1,3)<0)
        Ngamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3))+pi;
    else
        Ngamma(i+1)=atan(-Ctb(1,3)/Ctb(3,3))-pi;
    end
end
Npsi(1,lengthStill+1)=75 /180*pi;


    %% 统一绘图
ts=linspace(0,(lengthStill-1)/500,lengthStill);

td=linspace((lengthStill)/500,(lengthTotalData-1)/500,lengthDynamic);



figure, hold on;
plot(Lambda(lengthStill+1:lengthTotalData)*180/pi,L(lengthStill+1:lengthTotalData)*180/pi,'b
');
plot(NLambda(lengthStill+1:lengthTotalData)*180/pi,NL(lengthStill+1:lengthTotalData)*180/pi,
'r');
title('经纬度位置曲线'); xlabel('经度/° '); ylabel('纬度/° '); legend('先解算','先转化
','Location','southeast');
```

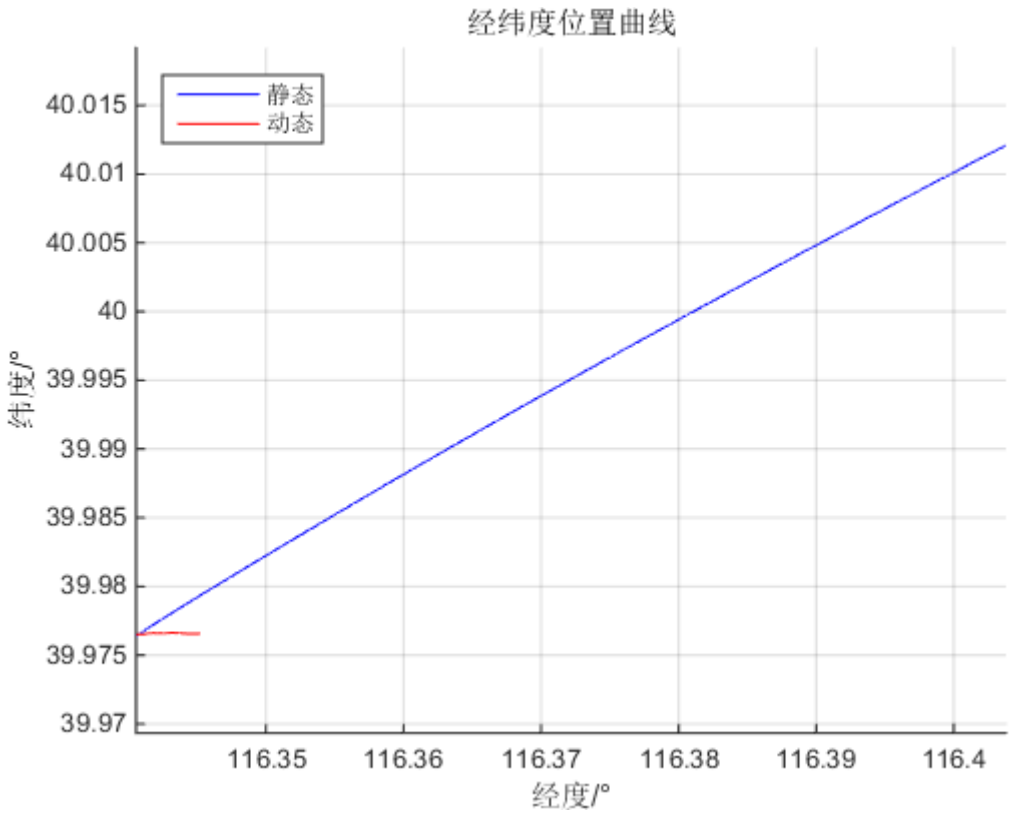
```
grid on; axis equal; hold off;

figure, hold on;
plot(td,Vx(lengthStill+1:lengthTotalData),'b');
plot(td,NVx(lengthStill+1:lengthTotalData),'r');
title('东西方向速度');xlabel('时间/s');ylabel('速度/m/s');legend('先解算','先转化
','Location','northwest');
grid on; hold off;

figure, hold on;
plot(td,Vy(lengthStill+1:lengthTotalData),'b');
plot(td,NVy(lengthStill+1:lengthTotalData),'r');
title('南北方向速度');xlabel('时间/s');ylabel('速度/m/s');legend('先解算','先转化
','Location','southwest');
grid on; hold off;

figure, hold on;
plot(td,base_theta(lengthStill+1:lengthTotalData)*180/pi,'b');
plot(td,Ntheta(lengthStill+1:lengthTotalData)*180/pi,'r');
title('俯仰角');xlabel('时间/s');ylabel('度');legend('先解算','先转化
','Location','northwest');
grid on; hold off;

figure, hold on;
plot(td,base_gamma(lengthStill+1:lengthTotalData)*180/pi,'b');
plot(td,Ngamma(lengthStill+1:lengthTotalData)*180/pi,'r');
title('滚转角');xlabel('时间/s');ylabel('度');legend('先解算','先转化
','Location','southwest');
grid on; hold off;

figure, hold on;
plot(td,base_psi(lengthStill+1:lengthTotalData)*180/pi,'b');
plot(td,Npsi(lengthStill+1:lengthTotalData)*180/pi,'r');
title('偏航角');xlabel('时间/s');ylabel('度');legend('先解算','先转化
','Location','northeast');
grid on; hold off;
```

五、实验结果及结论分析

1、静态与旋转调制的对比

（exp2_1.m 的运行结果与分析）

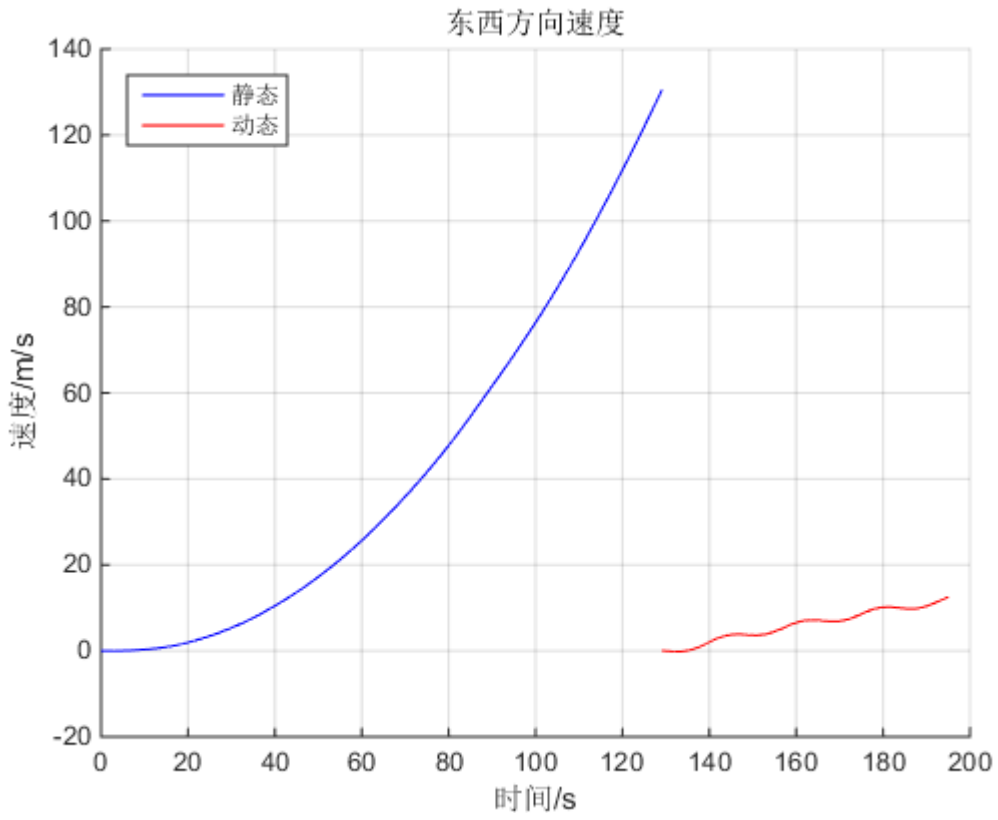
1）经纬度位置曲线

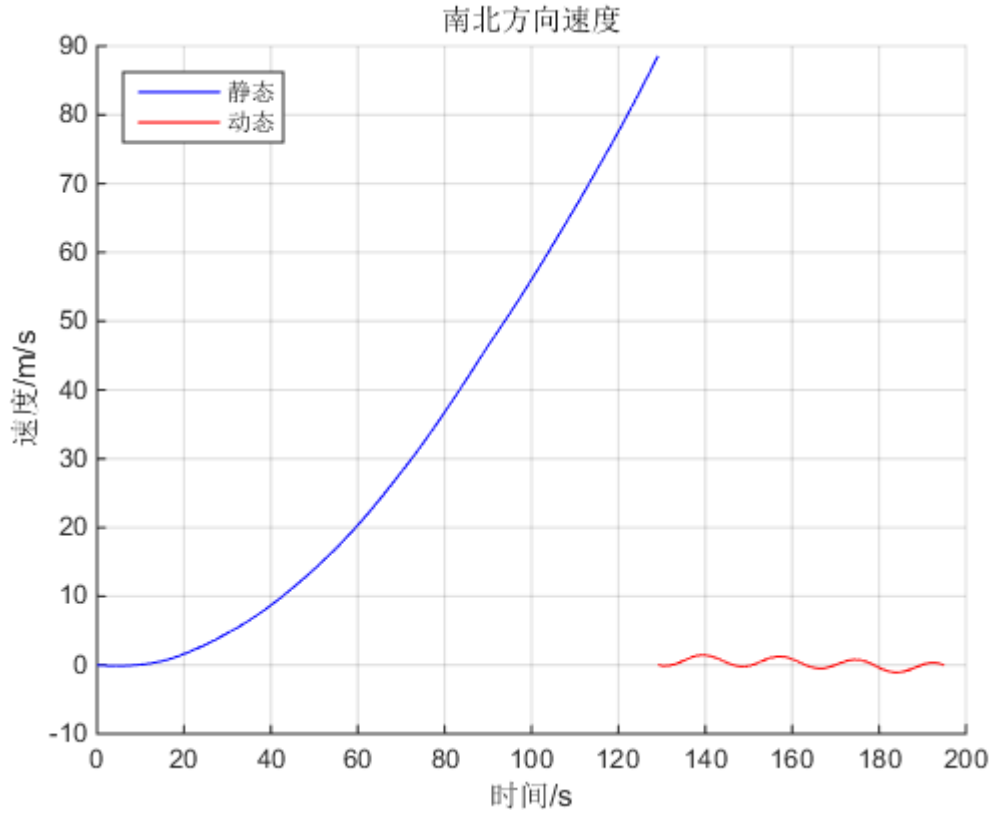


两段数据采集时间相差并不大，但是静态位置漂移巨大（蓝线），而经过旋转调制的数据（红线）位置漂移相对很小。

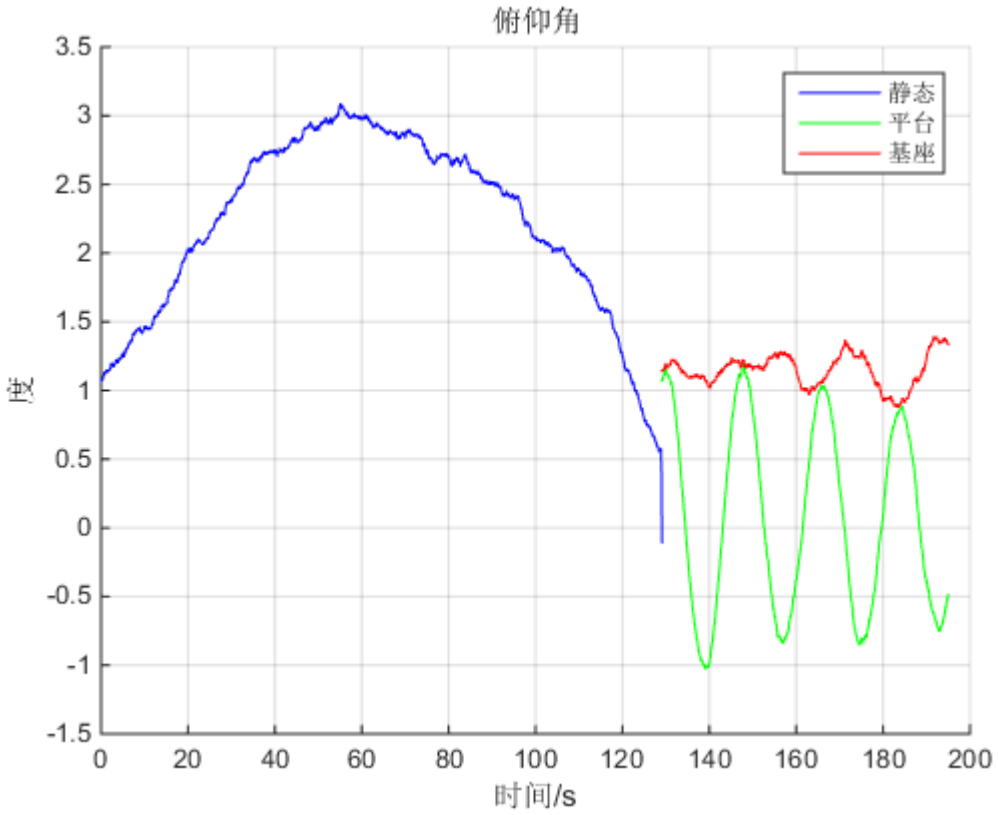
2）速度曲线

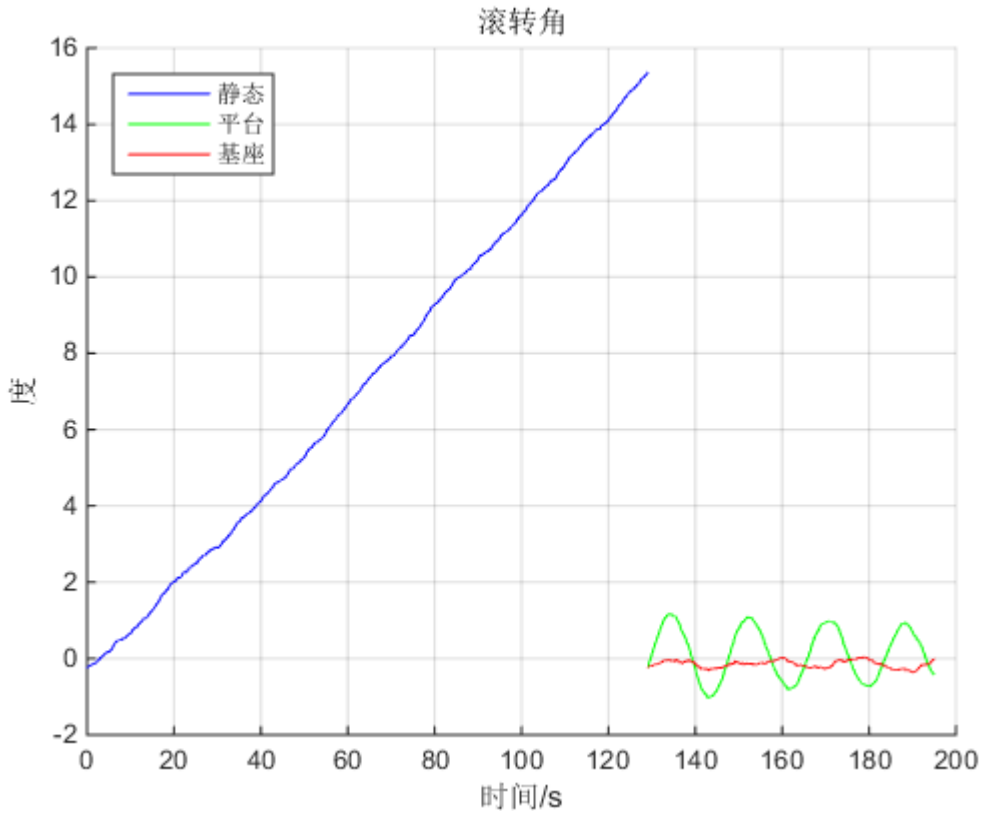
和之前的结论相同，经过旋转调制后的导航参数的漂移比静态的导航参数的漂移要小很多，这里静态的速度由于结算过程中的积分作用漂移到近 100m/s，而旋转调制的数据在结算过程中由于旋转调制的作用，有效地对积分作用产生的累计误差进行了抑制，从而得到的结果漂移较小。

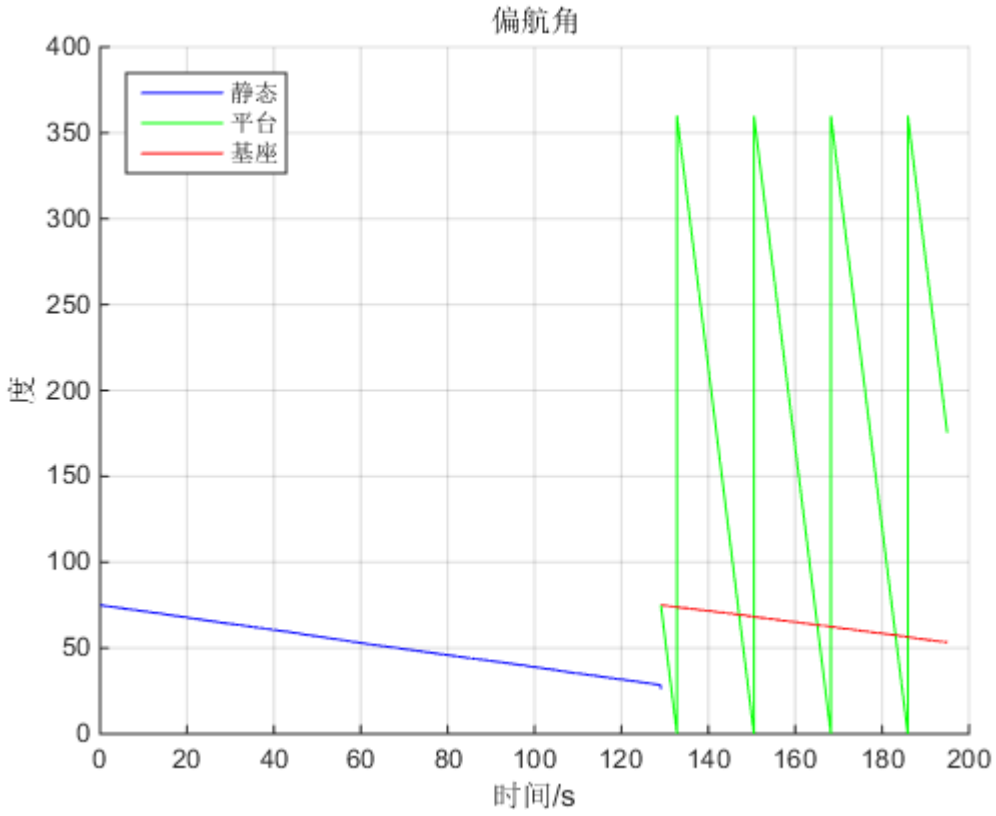

东西方向速度
静态
动态
速度/m/s
时间/s
140
120
100
80
60
40
20
0
-20
0
20
40
60
80
100
120
140
160
180
200


南北方向速度
静态
动态
速度/m/s
时间/s
90
80
70
60
50
40
30
20
10
0
-10
0
20
40
60
80
100
120
140
160
180
200

3）三个角度曲线







从俯仰角和滚转角的结果上看，经过旋转调制后解散出的导航参数比不经旋转调制（静态）解算得到的导航参数的漂移少很多。

以上三张结果图中的绿线都是 IMU 的导航参数，由于旋转调制，可以看到它们都呈周期性变化；从 IMU（和转台）偏航角结果更可以看出它是以恒定的角速度再旋转的。

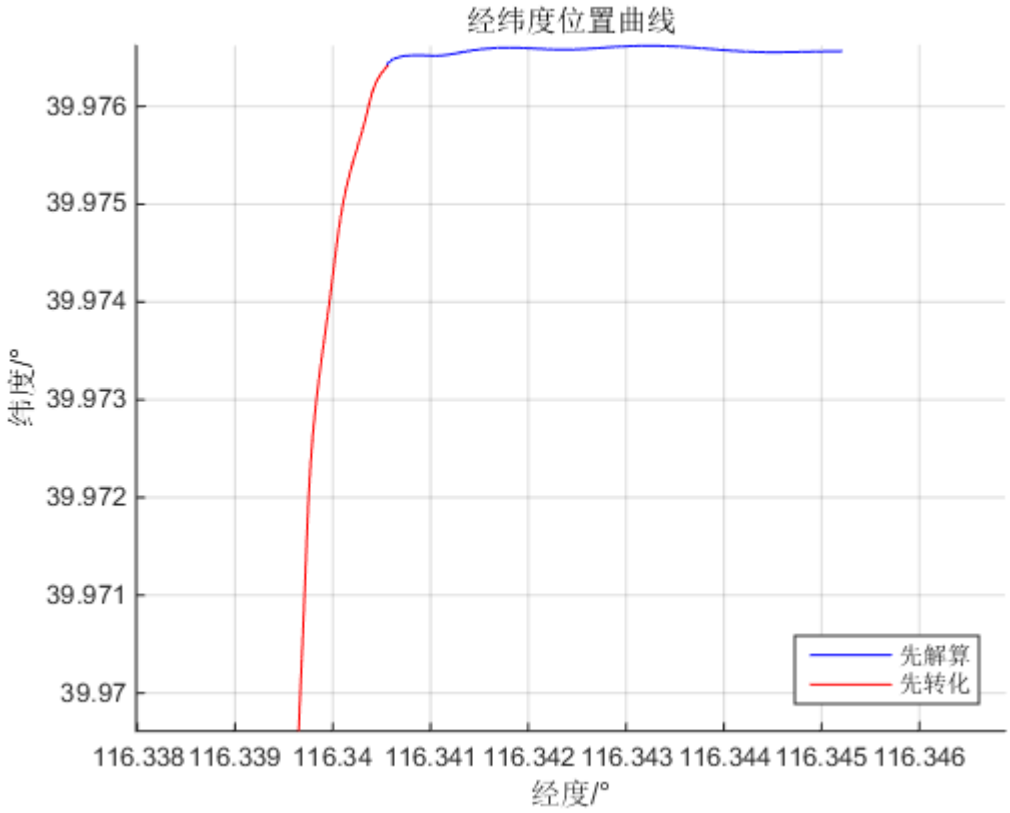
以上三张结果图中的红线表示的是最终转化得到的基座的导航参数，其中的俯仰角和滚转角经过安装误差修正后效果很好。

2、“先解算再转化”与“先转化再解算”的对比

（exp2_2.m 的运行结果与分析）

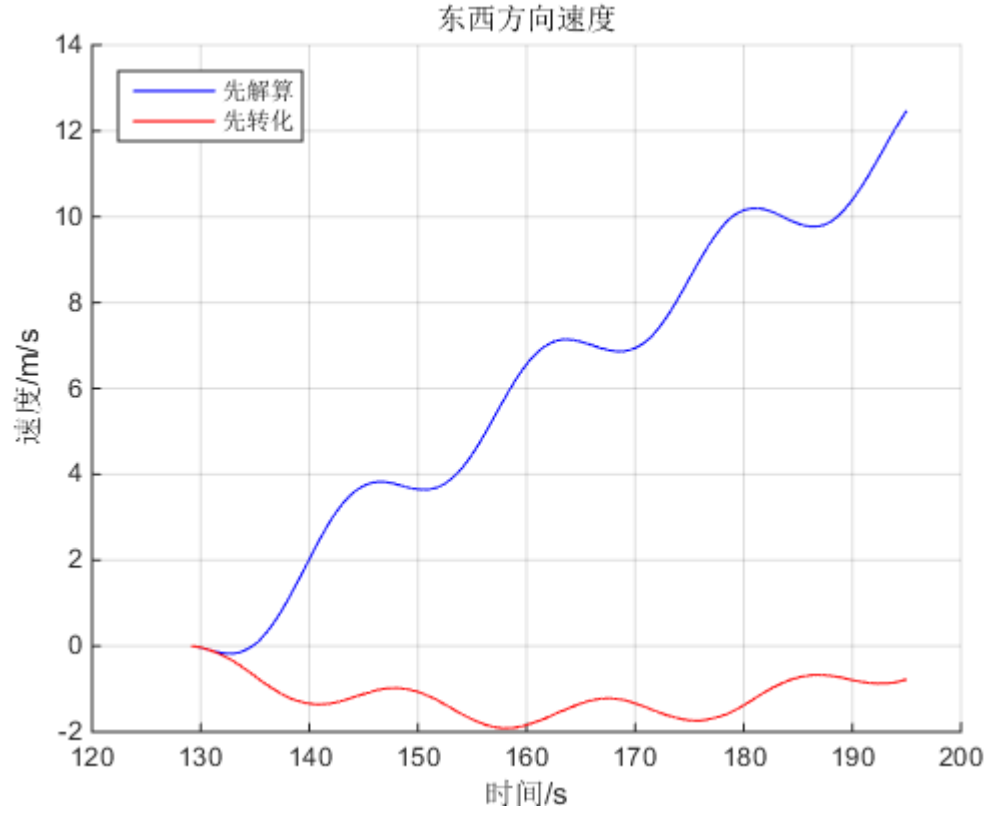
在第二个程序中先后采用了两种不同的方法对旋转调制部分的导航参数进行了处理，分别是先解算出 IMU 的导航参数在经过转化得到基座的导航参数；和先将 IMU 测得的数据转化到基座系在解算出基座的导航参数。以下为结果：

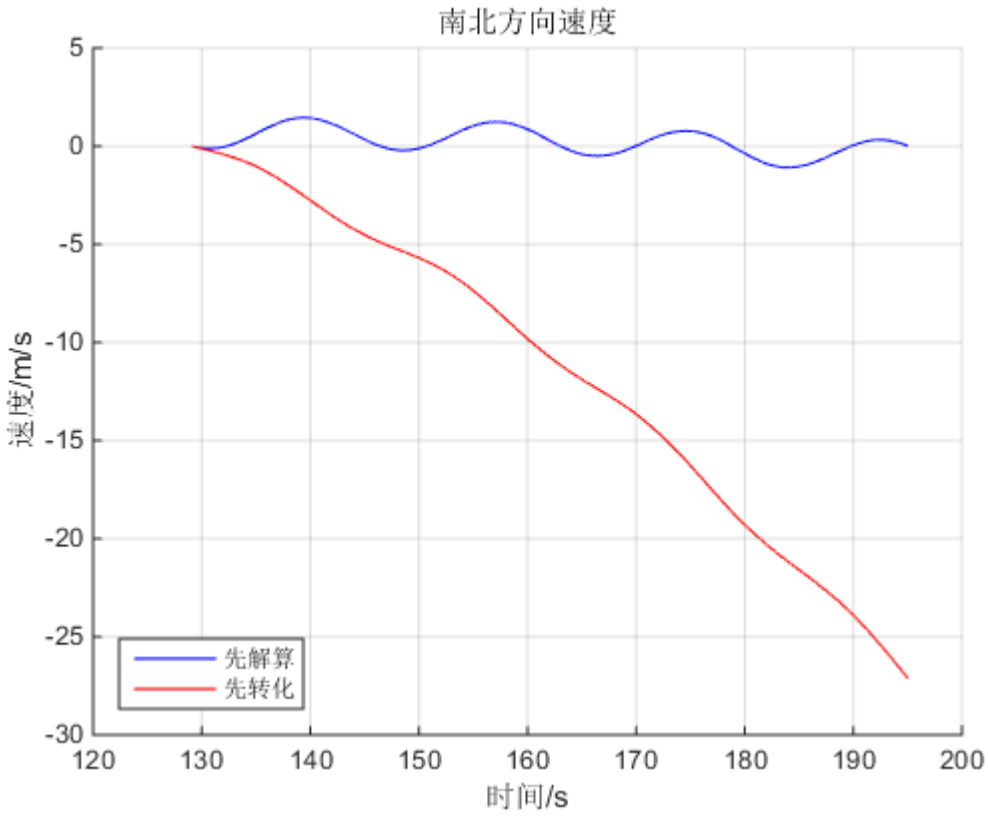
1）经纬度位置曲线



两种方法得到的位置解算结果误差漂移相当。

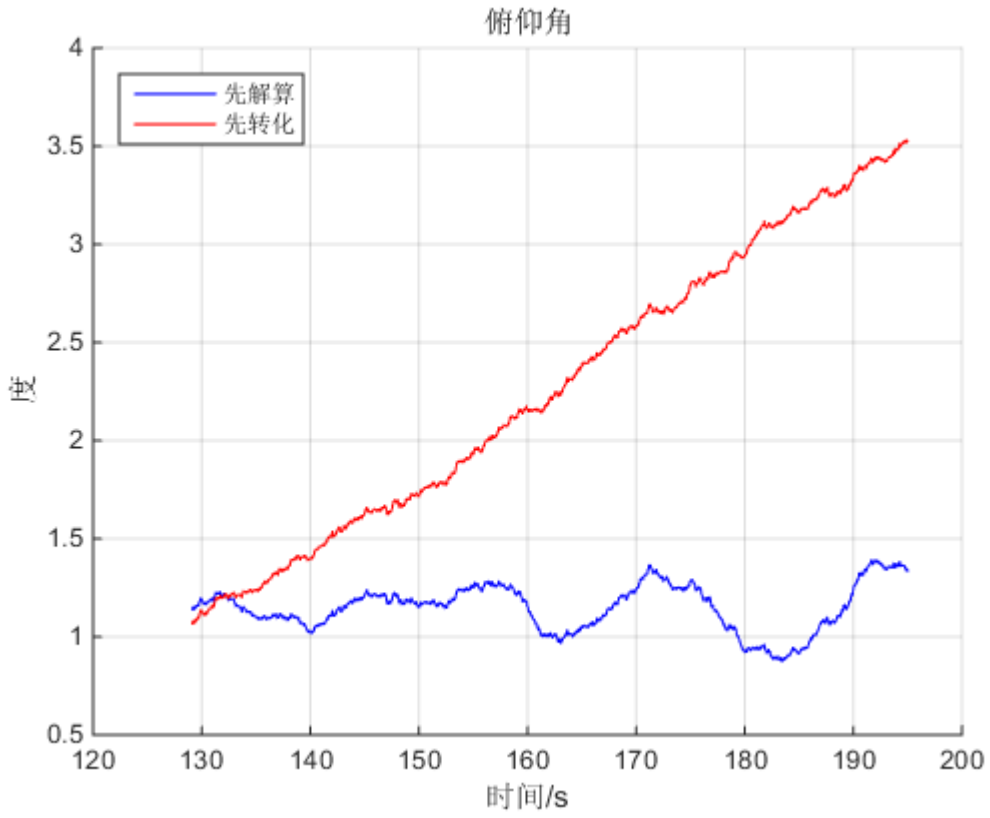
2）速度曲线

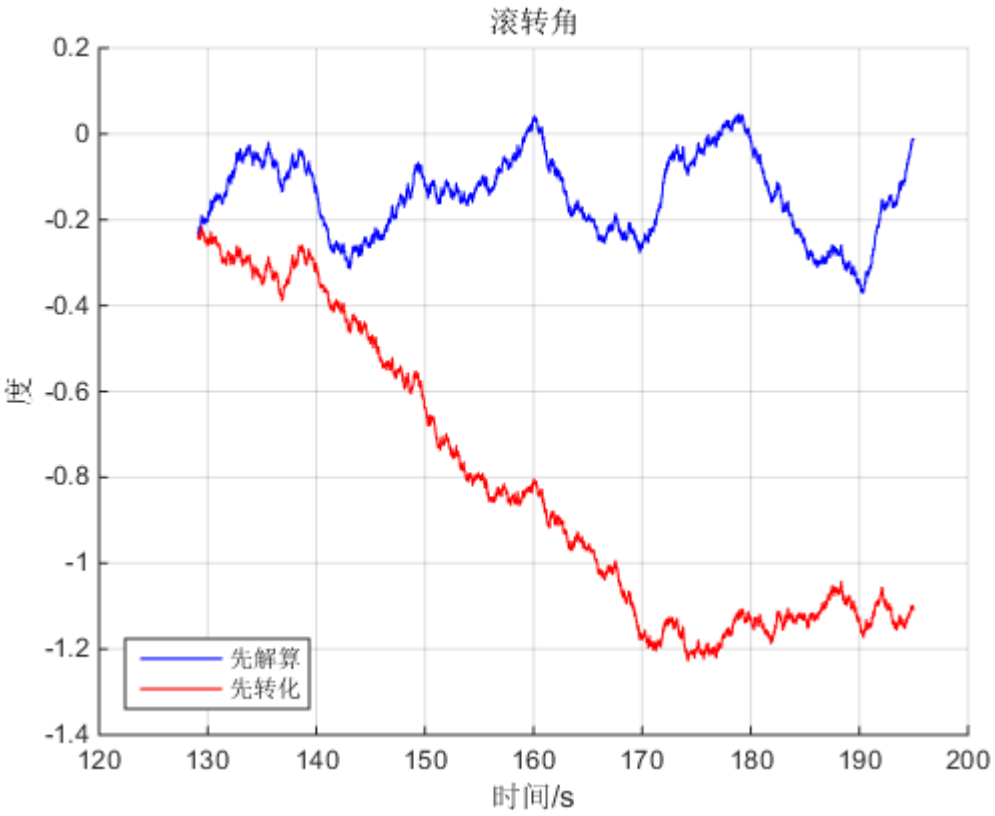


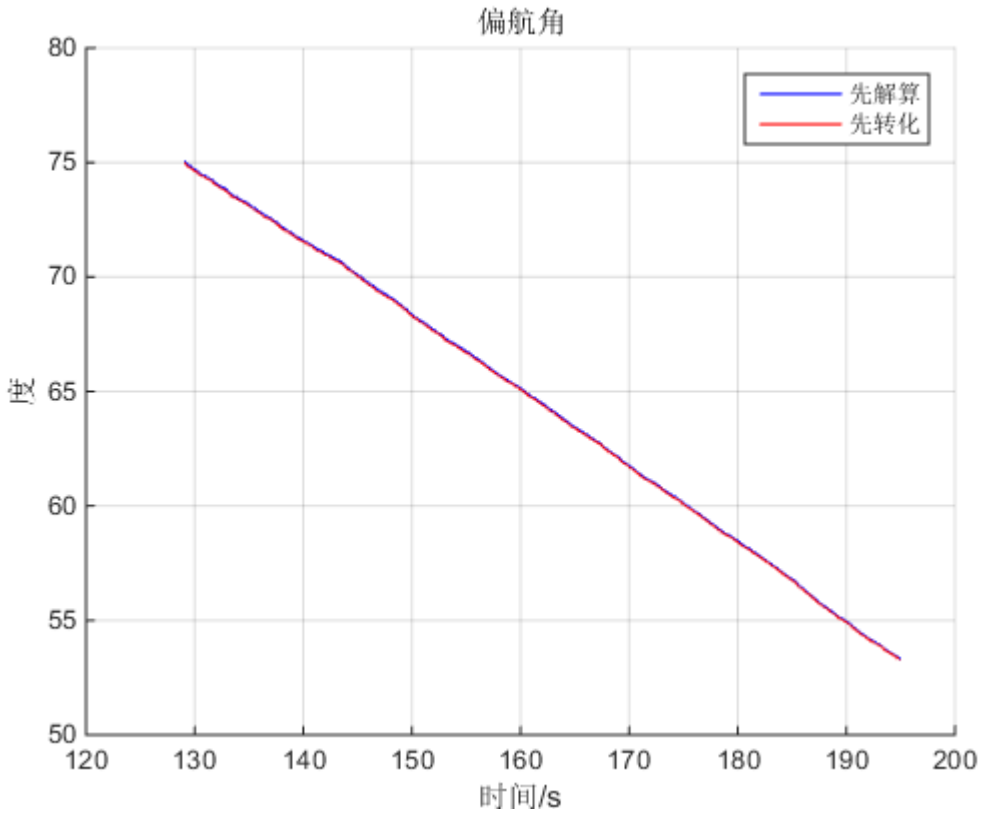


速度解算结果上看，先解算在转换在南北方向上的速度漂移比先转换再解算的结果要小，但是在东西方向上的速度漂移要大。总体上来看这两种方法得到的速度结果漂移情况相当，尚不清楚产生这种现象的原因。

3）三个角度曲线


俯仰角
先解算
先转化
4
3.5
3
2.5
2
1.5
1
0.5
120
130
140
150
160
170
180
190
200
时间/s


滚转角
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
先解算
先转化
120
130
140
150
160
170
180
190
200
时间/s



虽然在之前的速度结果的比较上两种方法的优势相当，但是在解算出的三个角度结果（滚转、俯仰、偏航）上可以看出先解算再转换得出的导航参数的漂移要比先转换再解算得到的结果的漂移小。

偏航角用两种方法得到的结果基本上无差异。

六、实验结论

本次实验对旋转调制进行了相对深入的探讨。

在惯导的过程中进行旋转调制可以有效的减小结果的漂移现象，以提高惯性导航的精度。旋转调制是通过对转台匀速旋转以使得 IMU 测得周期性的数据（比如这次试验中测得的 x 轴和 y 轴的比力），由于在计算过程中有积分环节的存在，对周期性数据进行积分时可以自然消除一部分误差的影响。

本次解算旋转调制部分数据的过程中还得到了“先对 IMU 导航参数解算再转化到基座系”的方法优于“先将 IMU 数据转化到基座系再进行导航解算”的方法，具体体现在前者的滚转角和偏航角的漂移要小于后者。

本次实验数据处理的不足：

对 IMU 测得数据进行观察，发现存在一定的噪声干扰。我在处理数据的时候曾尝试过先用其他课上学过的 IIR 或 FIR 滤波器进行滤波处理，想滤去噪声的干扰，但是发现用不用滤波器对于最终的结果影响并不显著，反而还大大增加了计算量。后来想用卡尔曼滤波处理，但是由于在一定时间内没有弄懂其原理，最终并没有实施这个方法。

对本实验的建议：

实验时若有足够的时间的话，要是能够指导我们自己测量 IMU 数据就更好了。

2016 年 5 月 14 日