```
clear all;
close all;
%% 文件读取
fid=fopen("11_25_18_CHA3_IMU.bin",'rb');
[IMU_DATA,N]=fread(fid,'double');
IMU_DATA=reshape(IMU_DATA,7,N/7).';
fclose(fid);
load('real.mat')
%% 初始化操作
IMU_DATA=IMU_DATA(17900:end,:);
Time0=IMU_DATA(64810,1);
omiga_e= 7.292115*1e-5;
Pos0=[30.5279001623000,114.355660640600,21.1180000000000];
Vel0=[0.0;0.0;0.0];
Start_time=find(IMU_DATA(:,1)==Time0);
Alig=IMU_DATA(1:Start_time,2:7)*200;
% Atti0=rad2deg(Find_Attitude(Alig,GRS80_g(Pos0),omiga_e,Pos0));
Atti0=[0.0358744800000000;0.724682280000000;1.67351167000000];
stop_time=[118560:129288,130514:132873,181340:195340,244690:258270,297710:310700];
%% 姿态、速度、位置更新
IMU_result=zeros(height(IMU_DATA)-Start_time,10);
E=zeros(height(IMU_DATA)-Start_time,3);
Pos=zeros(height(IMU_DATA)-Start_time,3);
Vel=zeros(height(IMU_DATA)-Start_time,3);
for i=1:height(IMU_DATA)-Start_time
    IMU_result(i,1)=IMU_DATA(Start_time+i,1);
    if find(stop_time==i+Start_time)
%         e=Euler2C(deg2rad(E(i-1,1:3)).');
%         v=[0,0,0].';
%         pos=Pos(i-1,1:3).';
        e=Update_Euler_C(deg2rad(E(i-1,:).'),IMU_DATA(Start_time+i-1,2:4).',IMU_DATA(St
        v=[0;0;0];
        pos=Update_pos(Pos(i-1,:).',Vel(i-1,:).',v,delt_t);
    else
        delt_t=IMU_DATA(Start_time+i,1)-IMU_DATA(Start_time+i-1,1);
        if i==1
            e=Update_Euler_C(deg2rad(Atti0),IMU_DATA(Start_time+i-1,2:4).',IMU_DATA(Sta
            v=Update_velocity(Vel0,Vel0,Pos0,Pos0,delt_t,IMU_DATA(Start_time+i-1,5:7).'
            pos=Update_pos(Pos0,Vel0,v,delt_t);
        elseif i==2
            e=Update_Euler_C(deg2rad(E(i-1,:).'),IMU_DATA(Start_time+i-1,2:4).',IMU_DAT
            v=Update_velocity(Vel0,Vel(i-1,1:3).',Pos0,Pos(i-1,1:3).',delt_t,IMU_DATA(S
            pos=Update_pos(Pos(i-1,:).',Vel(i-1,:).',v,delt_t);
        else
            e=Update_Euler_C(deg2rad(E(i-1,:).'),IMU_DATA(Start_time+i-1,2:4).',IMU_DAT
            v=Update_velocity(Vel(i-2,1:3).',Vel(i-1,1:3).',Pos(i-2,1:3).',Pos(i-1,:).'
            pos=Update_pos(Pos(i-1,:).',Vel(i-1,:).',v,delt_t);
        end
    end
    E(i,1:3)=rad2deg(C2Euler(e)).';
    Vel(i,1:3)=v.';
    Pos(i,1:3)=pos.';
end
```

```
IMU_result(:,2:10)=[Pos,Vel,E];
%% 转当地坐标系
NE=BLH2NE(Pos,Pos0);
NE_real=BLH2NE([real(:,3),real(:,2),real(:,4)],Pos0);
%%
draw_data(NE(:,2),NE(:,1),NE_real(:,2),NE_real(:,1))
%% 初始对准
function Euler=Find_Attitude(data,g,omiga_e,pos)
% 输入  data     原始观测数据[GRYO_X,GRYO_Y,GRYO_Z,f_X,f_Y,f_Z]    rad/s m·s^{-2}
%       g        重力加速度值                                      m·s^{-2}
%       omiga_e  地球自转角速度                                    rad/s
%       pos      初始位置                                          deg/deg/m
% 输出  Euler    初始姿态欧拉角                                    rad/rad/rad
    Lat=deg2rad(pos(1));
    g_n=[0;0;g];
    v_g=g_n/norm(g_n);
    omiga_n_ie=[omiga_e*cos(Lat);0;-omiga_e*sin(Lat)];
    v_omiga=cross(g_n,omiga_n_ie)/norm(cross(g_n,omiga_n_ie));
    v_gomiga=cross(cross(g_n,omiga_n_ie),g_n)/norm(cross(cross(g_n,omiga_n_ie),g_n));
    omiga_b_ie=[mean(data(:,1));mean(data(:,2));mean(data(:,3))];
    g_b=-[mean(data(:,4));mean(data(:,5));mean(data(:,6))];
    omiga_g=g_b/norm(g_b);
    omiga_omiga=cross(g_b,omiga_b_ie)/norm(cross(g_b,omiga_b_ie));
    omiga_gomiga=cross(cross(g_b,omiga_b_ie),g_b)/norm(cross(cross(g_b,omiga_b_ie),g_b)
    C_n_b=[v_g,v_omiga,v_gomiga]*[transpose(omiga_g);transpose(omiga_omiga);transpose(c
    Euler=C2Euler(C_n_b);
end
%% 线性外推函数
function result=Extrapol(v0,v1)
% 输入 v0      历元0数据
%      v1      历元1数据
% 输出 result 外推历元2数据
    result=1.5*v1-0.5*v0;
end
%% 当地重力值
function local_g=GRS80_g(pos)
% 输入 pos      位置         deg/deg/m
% 输出 local_g  当地重力值   m·s^{-2}
    B=deg2rad(pos(1));
    H=pos(3);
    g0=9.7803267715*(1+0.0052790414*sin(B)^2+0.0000232718*sin(B)^4);
    local_g=g0-(3.087691089*1e-6-4.397731*1e-9*sin(B)^2)*H+0.721*1e-12*H^2;
end
%% 子午圈半径
function R_M=Cal_RM(B) %#ok<*DEFNU>
% 输入 B    纬度         rad
% 输出 R_M  子午圈半径   m
    a=6378137.0;
    e=0.08181919104;
    R_M=a*(1-e^2)/sqrt((1-e^2*sin(B)^2)^3);
end
%% 卯酉圈半径
function R_N=Cal_RN(B)
```

```
% 输入 B     纬度          rad
% 输出 R_M   子午圈半径    m
    a=6378137.0;
    e=0.08181919104;
    R_N=a/sqrt(1-e^2*sin(B)^2);
end
%% 叉乘辅助矩阵
function Cross_Matrix=Cross_vector(vector)
    Cross_Matrix=[0,-vector(3),vector(2);
                  vector(3),0,-vector(1);
                  -vector(2),vector(1),0];
end
%% 四元数圈乘计算函数
function Multed_q=Mult_quaternion(q1,q2)
    M=eye(4,4)*q1(1);
    M(1,2:4)=-q1(2:4).';
    M(2:4,1)=q1(2:4);
    M(2:4,2:4)=M(2:4,2:4)+Cross_vector(q1(2:4));
    Multed_q=M*q2;
end
%% 四元数共轭
function conj_q=Conjugate(q)
    conj_q(1)=q(1);
    conj_q(2:4)=-q(2:4);
end
%% 四元数求逆
function inv_q=Inv_quaternion(q)
    inv_q=Conjugate(q)/norm(q)^2;
end
%% 姿态矩阵转四元数
function q=C2q(C)
    t=trace(C);
    P1=1+t;
    P2=1+2*C(1,1)-t;
    P3=1+2*C(2,2)-t;
    P4=1+2*C(3,3)-t;
    maxP=max([P1,P2,P3,P4]);
    if maxP==P1
        q1= 0.5*sqrt(P1);
        q2=(C(3,2)-C(2,3)/(4*q1));
        q3=(C(1,3)-C(3,1)/(4*q1));
        q4=(C(2,1)-C(1,2)/(4*q1));
    elseif maxP==P2
        q2= 0.5*sqrt(P2);
        q3=(C(2,1)+C(1,2)/(4*q2));
        q4=(C(1,3)+C(3,1)/(4*q2));
        q1=(C(3,2)-C(2,3)/(4*q2));
    elseif maxP==P3
        q3= 0.5*sqrt(P3);
        q2=(C(1,2)+C(2,1)/(4*q3));
        q1=(C(1,3)-C(3,1)/(4*q3));
        q4=(C(3,2)+C(2,3)/(4*q3));
    elseif maxP==P4
```

```
        q4= 0.5*sqrt(P4);
        q3=(C(3,2)+C(2,3)/(4*q4));
        q2=(C(1,3)+C(3,1)/(4*q4));
        q1=(C(2,1)-C(1,2)/(4*q4));
    end
    q=[q1;q2;q3;q4];
end
%% 四元数转姿态矩阵
function C=q2C(q)
    C=[q(1)^2+q(2)^2-q(3)^2-q(4)^2,2*(q(2)*q(3)-q(1)*q(4)),2*(q(2)*q(4)+q(1)*q(3));
       2*(q(2)*q(3)+q(1)*q(4)),q(1)^2-q(2)^2+q(3)^2-q(4)^2,2*(q(3)*q(4)-q(1)*q(2));
       2*(q(2)*q(4)-q(1)*q(3)),2*(q(3)*q(4)+q(1)*q(2)),q(1)^2-q(2)^2-q(3)^2+q(4)^2];
end
%% 四元数转等效旋转矢量
function Phi=q2Phi(q)
    if q(1)~=0
        half_phi=norm(q(2:4))/q(1);
        f=sin(half_phi)/(2*phi);
        Phi=q(2:4)/f;
    else
        Phi=pi*q(2:4);
    end
end
%% 等效旋转矢量转姿态矩阵
function C=Phi2C(Phi)
    n_phi=norm(Phi);
    C=eye(3,3)+sin(n_phi)/n_phi*Cross_vector(Phi)+(1-cos(n_phi))/(n_phi^2)*Cross_vector
end
%% 等效旋转矢量转四元数
function q=Phi2q(Phi)
    q=[cos(0.5*norm(Phi));sin(0.5*norm(Phi))/(0.5*norm(Phi))*0.5*Phi];
end
%% 姿态矩阵转欧拉角
function Euler=C2Euler(C)
    if abs(C(3,1))<0.999
        pitch=atan(-C(3,1)/sqrt((C(3,2)^2)+(C(3,3)^2)));      %俯仰角/rad
        roll=atan2(C(3,2),C(3,3));                            %横滚角/rad
        yaw=atan2(C(2,1),C(1,1));                             %航向角/rad
    else
        pitch=-1;
        roll=-1;
        yaw=-1;
    end
    Euler=[roll;pitch;yaw];
end
%% 欧拉角转姿态矩阵
function C=Euler2C(Euler)
    theta=Euler(2);
    phi=Euler(1);
    psi=Euler(3);
    C=[cos(theta)*cos(psi),-cos(phi)*sin(psi)+sin(phi)*sin(theta)*cos(psi),sin(phi)*sin
       cos(theta)*sin(psi),cos(phi)*cos(psi)+sin(phi)*sin(theta)*sin(psi),-sin(phi)*cos
       -sin(theta),sin(phi)*cos(theta),cos(phi)*cos(theta)];
```

```
end
%% 欧拉角转四元数
function q=Euler2q(Euler)
    theta=Euler(2)*0.5;
    phi=Euler(1)*0.5;
    psi=Euler(3)*0.5;
    q=[cos(phi)*cos(theta)*cos(psi)+sin(phi)*sin(theta)*sin(psi);
       sin(phi)*cos(theta)*cos(psi)-cos(phi)*sin(theta)*sin(psi);
       cos(phi)*sin(theta)*cos(psi)+sin(phi)*cos(theta)*sin(psi);
       cos(phi)*cos(theta)*sin(psi)-sin(phi)*sin(theta)*cos(psi)];
end
%% 姿态更新
function q=Update_Euler_q(E,theta0,theta1,pos,v,dt)
% 输入 E           姿态欧拉角            rad rad rad
%      theta0      前一历元观测角增量    rad rad rad
%      theta1      当前历元观测角增量    rad rad rad
%      pos         前一历元坐标          deg deg m
%      v           前一历元速度          m/s m/s m/s
%      dt          与前一历元时间差      s
% 输出 q           当前历元姿态四元数
    omiga_e=7.292115*1e-5;
    B=deg2rad(pos(1));
    omiga_ie=[omiga_e*cos(B);0;-omiga_e*sin(B)];
    omiga_en=[v(2)/(Cal_RN(B)+pos(3));-v(1)/(Cal_RM(B)+pos(3));-v(2)*tan(B)/(Cal_RN(B)+
    phi_k=theta1+cross(theta0,theta1)/12;
    zeta=(omiga_en+omiga_ie)*dt;
    q_bb=Phi2q(phi_k);
    q_nn=Conjugate(Phi2q(zeta));
    q0=Euler2q(E);
    q=Mult_quaternion(Mult_quaternion(q_nn,q0),q_bb);
end
function C=Update_Euler_C(E,theta0,theta1,pos,v,dt)
% 输入 E           姿态欧拉角            rad rad rad
%      theta0      前一历元观测角增量    rad rad rad
%      theta1      当前历元观测角增量    rad rad rad
%      pos         前一历元坐标          deg deg m
%      v           前一历元速度          m/s m/s m/s
%      dt          与前一历元时间差      s
% 输出 C           当前历元姿态矩阵
    omiga_e=7.292115*1e-5;
    B=deg2rad(pos(1));
    omiga_ie=[omiga_e*cos(B);0;-omiga_e*sin(B)];
    omiga_en=[v(2)/(Cal_RN(B)+pos(3));-v(1)/(Cal_RM(B)+pos(3));-v(2)*tan(B)/(Cal_RN(B)+
    phi_k=theta1+cross(theta0,theta1)/12;
    zeta=(omiga_en+omiga_ie)*dt;
    C_bb=Phi2C(phi_k);
    C_nn=Phi2C(-zeta);
    C0=Euler2C(E);
    C=C_nn*C0*C_bb;
end
%% 计算重力/哥氏积分项
function dv_g_cor=Cal_deltv_g_cor(g,omiga_ie,omiga_en,v0,dt)
% 输入 g           重力加速度          m/s^{-2}
```

```
%       omiga_ie    地球自转速度       rad/s
%       omiga_en    地球自转速度       rad/s
%       v0          外推速度           m/s
%       dt          历元间时间差       s
% 输出 dv_g_cor    重力/哥氏积分项    m/s
    dv_g_cor=(g-cross((2*omiga_ie+omiga_en),v0))*dt;
end
%% 计算比力积分项
function dv_f=Cal_deltv_f(omiga_ie,omiga_en,C,theta0,theta1,v0,v1,dt)
% 输入 omiga_ie    地球自转速度               rad/s
%       omiga_en    地球自转速度               rad/s
%       theta0      前一历元观测角增量         rad rad rad
%       theta1      当前历元观测角增量         rad rad rad
%       v0          前一历元观测速度增量       m/s
%       v1          当前历元观测速度增量       m/s
%       dt          历元间时间差               s
% 输出 dv_f        比力积分项                 m/s
    zeta=(omiga_ie+omiga_en)*dt;
    dv_f0=v1+cross(theta1,v1)/2+(cross(theta0,v1)+cross(v0,theta1))/12;
    dv_f=(eye(3,3)-0.5*Cross_vector(zeta))*C*dv_f0;
end
%% 速度更新
function v=Update_velocity(v00,v0,pos,pos0,dt,dv0,dv1,C,theta0,theta1)
% 输入 v00         前两个历元速度             m/s
%       v0          前一历元速度               m/s
%       pos         前一个历元位置             deg deg m
%       pos0        前两个历元位置             deg deg m
%       dv0         前一历元观测速度增量       m/s
%       dv1         前一历元观测速度增量       m/s
%       dt          历元间时间差               s
%       C           当前历元姿态矩阵
%       theta0      前一历元观测角增量         rad rad rad
%       theta1      当前历元观测角增量         rad rad rad
% 输出 v           当前历元速度               m/s
    omiga_e=7.292115*1e-5;
    B=deg2rad(pos(1));
    omiga_ie=[omiga_e*cos(B);0;-omiga_e*sin(B)];
    omiga_en=[v0(2)/(Cal_RN(B)+pos(3));-v0(1)/(Cal_RM(B)+pos(3));-v0(2)*tan(B)/(Cal_RN(
    g=[0;0;GRS80_g(pos)];
    B0=deg2rad(pos0(1));
    omiga_ie0=[omiga_e*cos(B0);0;-omiga_e*sin(B0)];
    omiga_en0=[v00(2)/(Cal_RN(B0)+pos(3));-v00(1)/(Cal_RM(B0)+pos(3));-v00(2)*tan(B0)/(
    g0=[0;0;GRS80_g(pos0)];
    g=Extrapol(g0,g);
    omiga_ie=Extrapol(omiga_ie0,omiga_ie);
    omiga_en=Extrapol(omiga_en0,omiga_en);
    dv_g_cor=Cal_deltv_g_cor(g,omiga_ie,omiga_en,Extrapol(v00,v0),dt);
    dv_f=Cal_deltv_f(omiga_ie,omiga_en,C,theta0,theta1,dv0,dv1,dt);
    v=v0+dv_f+dv_g_cor;
end
%% 位置更新
function pos=Update_pos(pos0,v0,v1,dt)
% 输入 pos0    前一历元坐标     deg deg m
```

```
%       v0      前一历元速度     m/s
%       v1      当前历元速度     m/s
%       dt      历元间时间差     s
% 输出 pos      当前历元位置    deg deg m
    h=pos0(3)-(v0(3)+v1(3))/2*dt;
    meanh=(h+pos0(3))/2;
    B=deg2rad(pos0(1))+(v0(1)+v1(1))/(2*(Cal_RM(deg2rad(pos0(1)))+meanh))*dt;
    meanB=(deg2rad(pos0(1))+B)/2;
    L=deg2rad(pos0(2))+(v0(2)+v1(2))/((2*(Cal_RN(meanB))+meanh)*cos(meanB))*dt;
    pos=[rad2deg(B),rad2deg(L),h];
end
%% 转NE坐标系
function NE=BLH2NE(BLH,BLH0)
% 输入 BLH    待转换大地坐标     deg deg m
%       BLH0   中心坐标          deg deg m
% 输出 NE     转换后NE坐标       m m
    B=deg2rad(BLH0(1,1));
    L=deg2rad(BLH0(1,2));
    h=BLH0(1,3);
    R_M=Cal_RM(B);
    R_N=Cal_RN(B);
    NE=zeros(height(BLH),2);
    for i=1:height(BLH)
        NE(i,1)=(deg2rad(BLH(i,1))-B)*(R_M+h);
        NE(i,2)=(deg2rad(BLH(i,2))-L)*(R_N+h)*cos(B);
    end
end
```