# 惯导解算

## inertial\_solve\_rate.m

速率式惯导解算原理程序。

qvp = RK4(@ins\_qvp\_n, qvp, dt, [gyro0;acc0],[gyro1;acc1],[gyro2;acc2]);

qvp(1:4) = quatnormalize(qvp(1:4)')'; %quaternion normalization

一起解整个惯导解算微分方程组，包含10个方程。

qvp = ins\_qvp\_n3(qvp, dt, [gyro0;acc0],[gyro1;acc1],[gyro2;acc2]);

姿态、速度、位置解算分开进行，与报告相同。

qvp(10) = traj(k+1,4); %height locking

高度锁定。

## ins\_qvp\_n3.m

三步惯导解算子程序。

if norm(dv)>1e-10

if norm(dp)>1e-10

用来限制是否增加小的速度位置增量。

## inertial\_solve\_delta.m

增量式惯导解算原理程序。

算导航系旋转角速度、子午圈半径、卯酉圈半径时用的定位信息可以用外推值，也可以用上一时刻值。

p(3) = traj(k+1,4); %height locking

高度锁定。

## C\_test\_inertial\_solve.m

测试惯导解算C代码程序，运行结果与matlab原理程序相同，验证转C是否正确。执行速率式解算时使用imu = imu\_rate;，执行增量式解算时使用imu = imu\_delta;，还需要对test\_inertial\_solve.c进行更改。

## test\_inertial\_solve.c

测试惯导解算C代码时，C语言与matlab的接口程序。

state = inertial\_slove\_rate(qvp, imu, dt);

state = inertial\_slove\_delta(qvp, imu, dt);

速率式、增量式切换。

qvp[9] = h[k]; //height locking

高度锁定。

因为matlab的原因，重新运行程序static变量不会被重新赋值，所以将flag在接口程序中赋值。

## inertial\_solve.c

惯导解算算法C语言程序，包含速率式、增量式、零偏补偿。

# 静对准

## static\_alignment.m

模型1：速度、姿态

模型2：速度、姿态、北向地向陀螺仪零偏

模型3：速度、姿态、三轴陀螺仪零偏（航向不动）

模型4：速度、姿态、北向地向陀螺仪零偏、地向加速度计零偏（实际用）

模型5：速度、姿态、北向陀螺仪零偏（看只估快的好不好使）

注释掉的画图是写报告用的。

## static\_alignment\_theoretical.m

给定三轴陀螺仪、加速度计零偏值，计算使用静对准模型计算的理论估计值，6项分别为航向角、俯仰角、滚转角（度）、北向陀螺仪零偏、地向陀螺仪零偏（度/小时）、地向加速度计零偏（mg）

lat设置纬度，Cnb设置真实姿态、dgyro设置陀螺仪零偏（度/小时），dacc设置加速度计零偏。

# 传递对准

## transfer\_alignment.m

模型1：姿态、速度、位置

模型2：姿态、速度、位置、安装误差角

模型3：姿态、速度、位置、安装误差角、陀螺仪零偏

模型4：姿态、速度、位置、陀螺仪零偏

模型5：姿态、速度、位置、安装误差角、陀螺仪零偏、加速度计零偏

用system\_model = 'model\_5';切换模型。

p0\_error，v0\_error，att0\_error设置子惯导初始导航误差。

add\_noise.m添加噪声，分隔线下面的不要动。

## C\_test\_transfer\_alignment.m

测试传递对准C代码程序，运行结果与matlab原理程序相同，验证转C是否正确。仿真条件要与transfer\_alignment.m相一致。

1.2 Existing data用于对已有数据执行传递对准算法，需要注释掉1.1。

## test\_transfer\_alignment.c

测试传递对准C代码时，C语言与matlab的接口程序。

## transfer\_alignment.c

传递对准算法C语言程序，使用ransfer\_alignment.h中的ALIGNMENT\_MODLE宏切换模型，切换完使用compile\_command.m重新编译。