1. 在上一章基于地图定位的方法中，增加运动约束模型(不需要估计安装误差)，并：

1）对比添加运动约束前后，定位误差的变化，以组合惯导的输出为真值

2）由于 kitti 的组合惯导也有一定误差，因此对比时宜重点关注波动性，而不要过多纠结连续偏差

3）运动约束对参数和车的抖动较敏感，因此调参时需要多一些耐心，并设计一定机制，在车快速转向时不添加约束

4）若实在调试不出好的效果，可按折中方法，即只在输出时做约束，而不把约束加入滤波模型。

答：

（1）公式推导

由于 kitti 数据集中 b 系是前左上坐标系，因此运动约束模型与课件中略有区别，这里重新推导观测方程。

观测方程：

$$Y = G_t X + C_t N$$

$$Y = \left(\delta P^T, \delta V_{byz}{}^T, \emptyset^T\right)^T$$

其中$\delta V_{byz} = (\delta V_{by}, \delta V_{bz})^T$，表示速度误差在 b 系的 y 轴和 z 轴方向

$$G_t = \begin{pmatrix} I_{3\times3}, & 0_{3\times3}, & 0_{3\times3}, & 0_{3\times6} \\ 0_{2\times3}, & (C_n^b)_{yz}, & (-C_n^b V\times)_{yz}, & 0_{2\times6} \\ 0_{3\times3}, & 0_{3\times3}, & I_{3\times3}, & 0_{3\times6} \end{pmatrix}$$

$$C_t = I_{8\times8}$$

$$N = \left(n_{P_E}, n_{P_N}, n_{P_U}, n_{V_{by}}, n_{V_{bz}}, n_{\emptyset_E}, n_{\emptyset_N}, n_{\emptyset_U}\right)^T$$

（2）代码修改

首先在上一节课的基础上，添加 dv_noise 参数用于配置运动约束的噪声。为了支持无运动约束和有运动约束两种情况，通过判断参数 dv_noise 是否为正来添加运动约束，同时，由于车快速转弯时存在侧滑现象，因此通过 angular_rate_threshold 参数作为采用运动约束的角速度最大阈值，配置文件如下所示：

ros > src > lidar_localization > config > matching > ! kalman_filter.yaml > ...

```yaml
gyro_noise: 0.1
acc_noise: 0.1
dp_noise: 0.1
dphi_noise: 0.02
dv_noise: 0.01
angular_rate_threshold: 0.8
```

下一步主要修改了 lidar_localization/src/matching/localization_flow.cpp 文件。预测部分，在 Predict 函数中通过 IMU 惯性解算求得当前时刻的角速度，如下所示：

```cpp
Eigen::Quaterniond dq_imu = state_.q.inverse() * qq;
angular_rate_ = std::acos(std::fabs(dq_imu.w())) * 2
/ (current_imu_data_.back().time - current_imu_data_.front().time);
```

观测部分，在 Correct 函数中首先判断 dv_noise 和角速度是否满足添加运动约束的条件，如果没有，则不添加运动约束，即观测维度为 6，如下所示

```cpp
if (dv_noise_ <= 0 || angular_rate_ > angular_rate_threshold_)
{
    Eigen::Matrix<double, 6, 1> Y = Eigen::Matrix<double, 6, 1>::Zero();
    Y.head(3) = dp;
    Y.tail(3) = dphi;
```

否则，观测维度为 8，此时观测方程按（1）部分公式进行编写，如下所示

```
else
{
    Eigen::Matrix3d c_b_n = state_.q.toRotationMatrix().transpose();
    Eigen::Matrix<double, 8, 1> Y = Eigen::Matrix<double, 8, 1>::Zero();
    Y.head(3) = dp;
    Eigen::Vector3d dv = c_b_n * state_.v;
    Y(3) = dv(1);
    Y(4) = dv(2);
    Y.tail(3) = dphi;

    Eigen::Matrix<double, 8, 15> Gt = Eigen::Matrix<double, 8, 15>::Zero();
    Gt.block<3, 3>(0, 0) = Eigen::Matrix<double, 3, 3>::Identity();
    Gt.block<2, 3>(3, 3) = c_b_n.block<2, 3>(1, 0);
    Eigen::Matrix3d vnx = Eigen::Matrix3d::Zero();
    vnx(0, 1) = - state_.v(2);
    vnx(0, 2) = state_.v(1);
    vnx(1, 0) = state_.v(2);
    vnx(1, 2) = -state_.v(0);
    vnx(2, 0) = -state_.v(1);
    vnx(2, 1) = state_.v(0);
    Gt.block<2, 3>(3, 6) = (-c_b_n * vnx).block<2, 3>(1, 0);
    Gt.block<3, 3>(5, 6) = Eigen::Matrix<double, 3, 3>::Identity();

    Eigen::Matrix<double, 8, 8> Ct = Eigen::Matrix<double, 8, 8>::Identity();

    Eigen::Matrix<double, 8, 8> R = Eigen::Matrix<double, 8, 8>::Identity();
    R.block<3, 3>(0, 0) = Eigen::Matrix<double, 3, 3>::Identity() * dp_noise_ * dp_noise_;
    R.block<2, 2>(3, 3) = Eigen::Matrix<double, 2, 2>::Identity() * dv_noise_ * dv_noise_;
    R.block<3, 3>(5, 5) = Eigen::Matrix<double, 3, 3>::Identity() * dphi_noise_ * dphi_noise_;

    Eigen::Matrix<double, 15, 8> K = error_state_.p * Gt.transpose()
    * (Gt * error_state_.p * Gt.transpose() + Ct * R * Ct.transpose()).inverse();
    error_state_.p = (Eigen::Matrix<double, 15, 15>::Identity() - K * Gt) * error_state_.p;
    error_state_.x = error_state_.x + K * (Y - Gt * error_state_.x);
```

为了后续结果分析，需要保存轨迹和速度数据，因此添加以下两个函数

```
bool LocalizationFlow::SavePose(std::ofstream& ofs, const Eigen::Matrix4f& pose) {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 4; ++j) {
            ofs << pose(i, j);

            if (i == 2 && j == 3) {
                ofs << std::endl;
            } else {
                ofs << " ";
            }
        }
    }

    return true;
}

bool LocalizationFlow::SaveVelocity(std::ofstream& ofs, double stamp, const Eigen::Vector3d& vel)
{
    velocity_ofs_ << stamp << " "
    << vel[0] << " "
    << vel[1] << " "
    << vel[2] << std::endl;
    return true;
}
```

并在每次滤波后存入文件，如下所示

```
SavePose(ground_truth_ofs_, gnss_pose_);
SavePose(localization_ofs_, pose);
SaveVelocity(velocity_ofs_, current_cloud_data_.time, state_.q.inverse() * state_.v);
```

（3）运行结果与分析

a) 轨迹误差

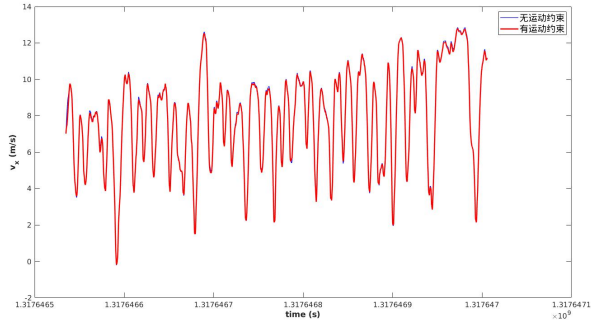
无运动约束和有运动约束的轨迹整体误差如下所示，根据对比，两者差别很小，但有运动约束的轨迹最大误差要稍微好于无运动约束的轨迹。

类型	无运动约束	有运动约束
误差结果	APE w.r.t. translation part (m) (not aligned) max 1.403499 mean 0.289208 median 0.233706 min 0.009211 rmse 0.349102 sse 553.665763 std 0.195527	APE w.r.t. translation part (m) (not aligned) max 1.327003 mean 0.292118 median 0.257054 min 0.017096 rmse 0.344137 sse 538.027542 std 0.181926
max		better

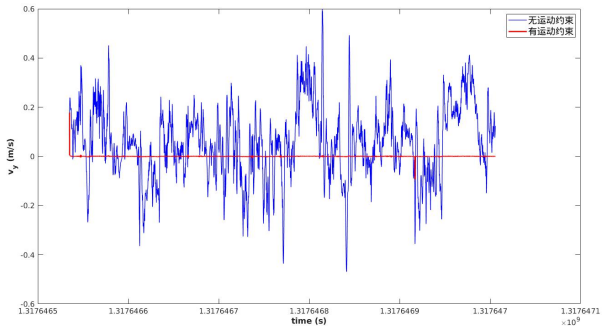
mean	better	
median	better	
min	better	
rmse		better
sse		better
std		better

b)　速度波动

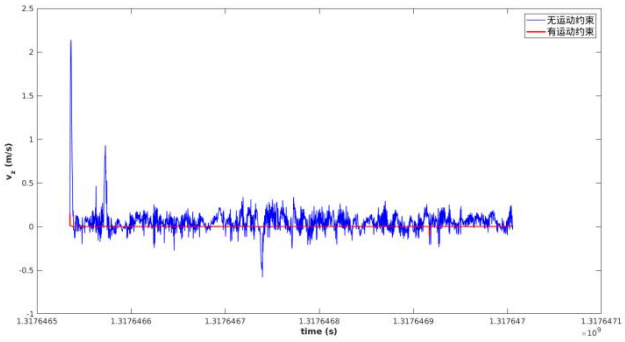
IMU 坐标系下 3 个轴的速度比较如下所示，可以看出，在车前进方向（x 轴）上两种方法差别很小，基本一致，而在车的侧向（y 轴）和竖直（z 轴）方向上，无运动约束的速度波动很大，大概在±0.2~0.3m/s 范围内波动，但添加了运动约束之后，速度基本保持在±0.01m/s 范围以内，符合噪声 dv_noise 等于 0.01m/s 的设定。



V_x



V_y



V_z

2. 在仿真数据中，增加里程计数据，实现“gps+imu+里程计”的滤波融合算法(不需估计安装误差)，并：

1）观察不同运动状态下的收敛情况，并与上一章的结果做对比。

2）选做：可自行拓展不同精度 imu、不同组合方式(比如不加 gps 观测)、不同模型(比如对比考虑地速和不考虑地速)下的滤波收敛情况。

答：

（1）公式推导

与课件相比，只有观测方程发生了变化，因此，这里只推导观测方程。

$$Y = G_t X + C_t N$$

$$Y = \left(\delta P^T, \delta {V_b}^T\right)^T$$

其中$\delta V_b = (\delta V_{bx}, \delta V_{by}, \delta V_{bz})^T$

$$G_t = \begin{pmatrix} I_{3\times3}, & 0_{3\times3}, & 0_{3\times3}, & 0_{3\times6} \\ 0_{3\times3}, & C_n^b, & -C_n^b V\times, & 0_{3\times6} \end{pmatrix}$$

$$C_t = I_{6\times6}$$

$$N = \left(n_{P_E}, n_{P_N}, n_{P_U}, n_{V_{bx}}, n_{V_{by}}, n_{V_{bz}}\right)^T$$

（2）代码实现

a) gnss-ins-sim 仿真

与上一章不同的地方是需要生成里程计数据，而车体坐标系规定为右前上，即里程计的数据应该是 y 轴的速度，而 gnss-in-sim 生成的是 x 轴，因此需要修改 gnss-ins-sim/gnss_ins_sim/sim/ins_sim.py，如下所示

```
if self.imu.odo:
    self.dmgr.add_data(self.dmgr.ref_odo.name, rtn['odo'][:, 3])
```

同时参数配置与上一章基本一致，只是添加了里程计的误差，如下所示

```
# imu_err
imu_err = {'gyro_b': np.array([1e-2, 2e-2, 3e-2]) / D2R * 3600 * 0,
           'gyro_arw': np.array([1e-5, 1e-5, 1e-5]) / D2R * 60 * 0,
           'gyro_b_stability': np.array([1e-5, 1e-5, 1e-5]) / D2R * 3600 * 1e-0,
             'gyro_b_corr': np.array([100.0, 100.0, 100.0]),
           'accel_b': np.array([2.0e-3, 1.0e-3, 5.0e-3]) * 0,
           'accel_vrw': np.array([1e-4, 1e-4, 1e-4]) * 60.0 * 0,
           'accel_b_stability': np.array([1e-4, 1e-4, 1e-4]) * 1.0 * 1e0,
             'accel_b_corr': np.array([200.0, 200.0, 200.0]),
             'mag_std': np.array([0.2, 0.2, 0.2]) * 0.0
           }
# generate GPS and magnetometer data
gps_err = {
    'stdp': np.array([1, 1, 1]) * 0e-6,
    'stdv': np.array([0, 0, 0]),
}
odo_err = {
    'scale': 1,
    'stdv': 0e-6,
}
imu = imu_model.IMU(accuracy=imu_err, axis=9, gps=True, gps_opt=gps_err, odo=True, odo_opt=odo_err)
```

b) 滤波

在上一章 ins 工程的基础上，添加里程计的观测部分。参数配置中添加了 odo_noise 作为里程计噪声，use_odom 来选择是否启用里程计观测，如下所示

```
init_noise: [1e-4, 1e-4, 1e-4, 1e-4, 1e-4]
init_dx: [0e-4, 0e-4, 0e-4, 0e-2, 0e-2]
gyro_noise: 1e-5
acc_noise: 1e-4
dp_noise: 1e-6
odo_noise: 1e-6
mag_noise: 1e-6
end_time: 5000.0
FGsize: 20
time_interval: 1
correct: true
use_odom: true
use_mag: false
```

在 Correct 函数中添加里程计观测，如下所示

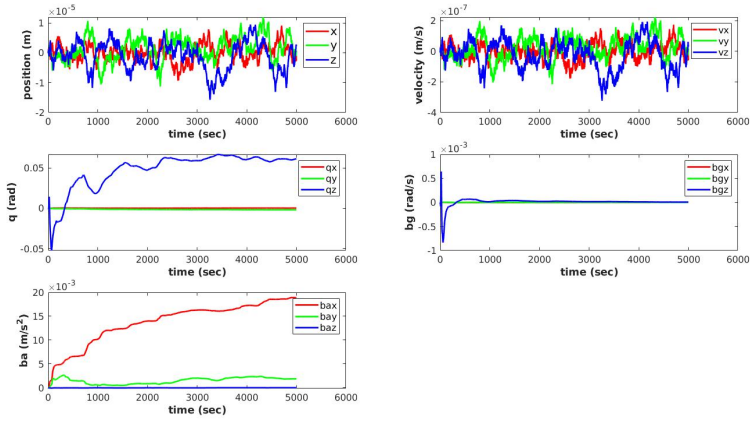
```
else if (use_odom)
{
    Eigen::Matrix3d cbn = current_pose.state.q.toRotationMatrix().transpose();
    Y.resize(6, 1);
    Y.block<3, 1>(0, 0) = current_pose.state.p - gnss_xyz;
    Y.block<3, 1>(3, 0) = cbn * current_pose.state.v - Eigen::Vector3d(0, current_odo.odo, 0);

    Gt = Eigen::Matrix<double, 6, 15>::Zero();
    Gt.block<3, 3>(0, 0) = Eigen::Matrix<double, 3, 3>::Identity();
    Gt.block<3, 3>(3, 3) = cbn;
    Eigen::Matrix3d vnx = Eigen::Matrix3d::Zero();
    vnx(0, 1) = - current_pose.state.v(2);
    vnx(0, 2) = current_pose.state.v(1);
    vnx(1, 0) = current_pose.state.v(2);
    vnx(1, 2) = -current_pose.state.v(0);
    vnx(2, 0) = -current_pose.state.v(1);
    vnx(2, 1) = current_pose.state.v(0);
    Gt.block<3, 3>(3, 6) = -cbn * vnx;

    Ct = Eigen::Matrix<double, 6, 6>::Identity();
    R = Eigen::Matrix<double, 6, 6>::Identity();
    R.block<3, 3>(0, 0) = Eigen::Matrix3d::Identity() * dp_noise * dp_noise;
    R.block<3, 3>(3, 3) = Eigen::Matrix3d::Identity() * odo_noise * odo_noise;
}
```

（3）不同运动状态下的收敛情况

a)　静止状态

仿真命令参见附件文件 imu_def11.csv。各状态量的收敛情况见下图，可观测度系数见下表。从图中可看出，航向角，x 轴加速度零偏和 y 轴加速度零偏均未收敛，而这 3 个状态量对应的可观测度系数均非常小，因此 3 个状态量是不可观的。另外，z 轴角速度零偏的收敛速度很慢，它对应的可观测度也很低，在 6e-7 左右，由于其仍然能够收敛，因此认为它是可观的。

结论：静止状态下与不添加里程计的结果一致，存在 3 个维度不可观。



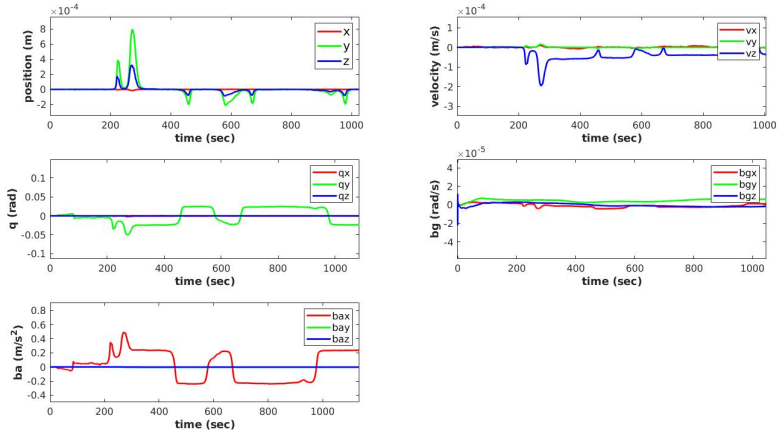
误差值（滤波值减去真实值）

可观测度	1	1	1	0.99 5037	0.99 5037	0.99 5037	0.98 4575	0.98 4575	0.1	0.09 7948 4	0.09 7948 4	6.04 E-07	1.85 E-08	1.98 E-09	7.49 E-18
状态量	5	4	3	0	1	2	6	7	14	9	10	11	8	12	13

b)　匀速状态

仿真命令参见附件文件 imu_def12.csv，b 系沿前向（y 轴正向）做匀速运动。各状态量的收敛情况见下图，可观测度系数见下表。从表中看出，可观测度矩阵的秩为 14，最小的可观测度系数对应的状态量是 x 轴加速度零偏。而从图中可看出，x 轴加速度零偏未收敛，但是除此之外，y 轴角度也发散了，这主要是由于两个状态量存在耦合关系，从图中可以看出，两者的变化规律有很明显的一致性。

结论：匀速状态下，不添加里程计有 3 个维度是不可观的，但是添加里程计后仅有 1 个维度是不可观的。



误差值（滤波值减去真实值）

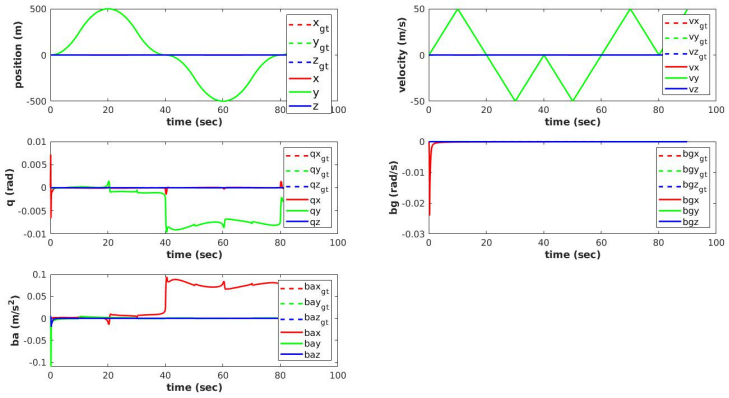
可观测度	1	0.96 3084	0.95 8398	0.95 3641	0.95 3641	0.95 3641	0.94 3662	0.90 4449	0.09 9822 5	0.09 3873 5	0.09 0132 7	9.44 E-03	9.64 E-04	9.49 E-04	2.78 E-17
状态量	5	3	4	0	1	2	7	6	14	10	9	8	13	11	12

c)　加减速状态

仿真命令参见附件文件 imu_def13.csv 和 imu_def14.csv ,两个运动命令分别对应了 b 系沿一个方向和两个方向做加减速运动，具体运动参考下图中的位置和速度变化曲线。可观测度系数见下表。沿一个方向运动时，收敛和可观测情况

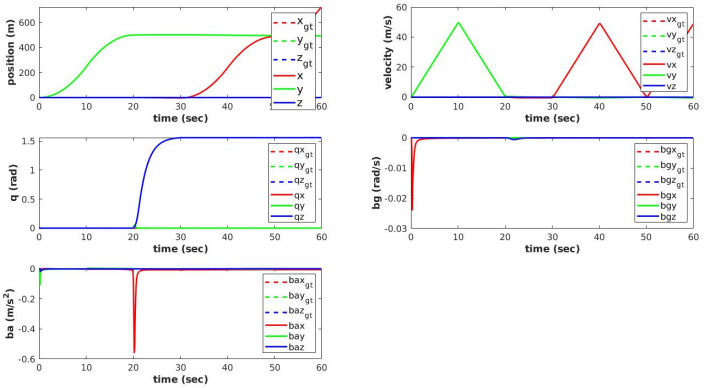
与匀速状态一致，存在 1 个维度是不可观的。而沿两个方向加减速，则所有状态量均收敛，且可观测系数均在 1e-3 量级以上，因此所有状态量均可观。

结论：沿一个方向做加减速时，不添加里程计和添加里程计均有 1 个维度不可观；沿两个方向做加减速时，不添加里程计和添加里程计均能保证所有状态量是可观的。



真实值与滤波值比较（imu_def13.csv）

可观测度	1	0.999374	0.100747	0.099992	0.036125	0.035942	0.035942	0.035942	0.039779	0.039777	0.031773	3.61E-03	3.54E-03	3.40E-03	1.70E-06
状态量	6	8	11	9	4	1	0	2	3	5	7	13	10	14	12



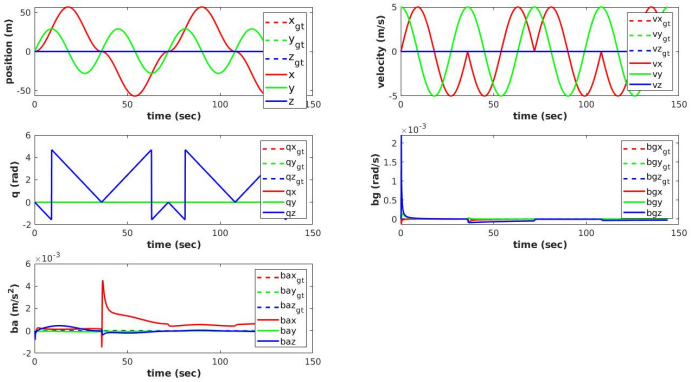
真实值与滤波值比较（imu_def14.csv）

可观测度	1	0.99917	0.104582	0.100005	0.042659	0.042447	0.042447	0.042447	0.030801	0.030225	0.026534	4.18E-03	4.06E-03	3.01E-03	2.30E-03
状态量	6	8	11	9	4	0	2	1	5	3	7	10	13	14	12

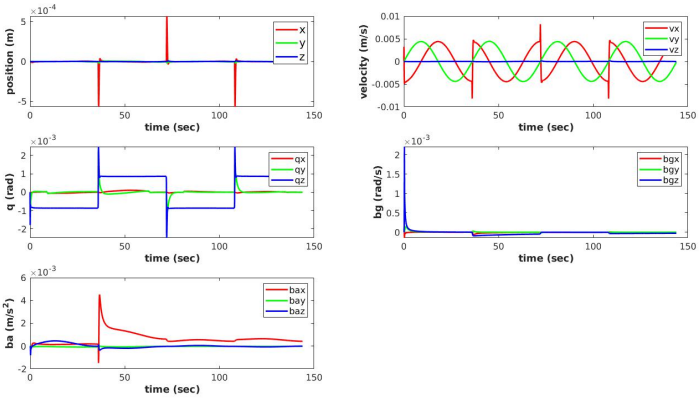
d) **转向状态**

仿真命令参见附件文件 imu_def15.csv，b 系做绕 8 字运动，具体运动情况见下图的位置和姿态变化。各状态量可观测度系数见下表。所有状态量均收敛，且可观测系数均在 1e-3 量级以上，因此所有状态量均可观。

结论：转向状态下，不添加里程计和添加里程计均能保证所有状态量是可观的。



真实值与滤波值比较



误差值（滤波值减去真实值）

可观测度	1	0.78 8511	0.67 6486	0.20 0814	0.19 9817	0.19 9817	0.19 9817	0.19 6976	0.19 1873	0.10 3714	0.10 1769	2.00 E-02	1.97 E-02	4.28 E-03	1.96 E-03
状态量	8	7	6	4	2	0	1	3	5	9	11	13	10	14	12

3. 在仿真数据中，增加磁力计数据，实现“gps+imu+磁力计”的滤波融合算法

答：

（1）公式推导

与课件相比，这里采用基于误差状态的方法进行滤波，而由于只有观测方程发生了变化，因此，这里只推导观测方程。

1）写出不考虑误差时的方程

$B_b = C_n^b B_n$

其中$B_n = (B_E, B_N, B_U)^T$

2）写出考虑误差时的方程

$\widetilde{B}_b = \widetilde{C}_n^b B_n$

3）写出真实值与理想值之间的关系

$\widetilde{B}_b = B_b + \delta B_b$

$\widetilde{C}_n^b = C_n^b(I + \emptyset\times)$

4）把 3)中的关系带入 2)式

$B_b + \delta B_b = C_n^b(I + \emptyset\times)B_n$

5）把 1)中的关系带入 4)式

$C_n^b B_n + \delta B_b = C_n^b(I + \emptyset\times)B_n$

6）化简方程

$\delta B_b = -C_n^b B_n \times \emptyset$

加入磁力计后，它的观测量可以写为

$$Y = \left(\delta P^T, \delta {B_b}^T\right)^T$$

其中δB_b的观测值可以通过下式获得

$\delta B_b = \widetilde{B}_b - B_b = \widetilde{C}_n^b B_n - B_b$

此时，可写出完整的观测方程，如下

其中

$$G_t = \begin{pmatrix} I_{3\times3}, & 0_{3\times3}, & 0_{3\times3}, & 0_{3\times6} \\ 0_{3\times3}, & 0_{3\times3}, -C_n^b B_n\times, & 0_{3\times6} \end{pmatrix}$$

$$C_t = I_{6\times6}$$

$$N = \left(n_{P_E}, n_{P_N}, n_{P_U}, n_{B_{bx}}, n_{B_{by}}, n_{B_{bz}}\right)^T$$

（2）代码实现

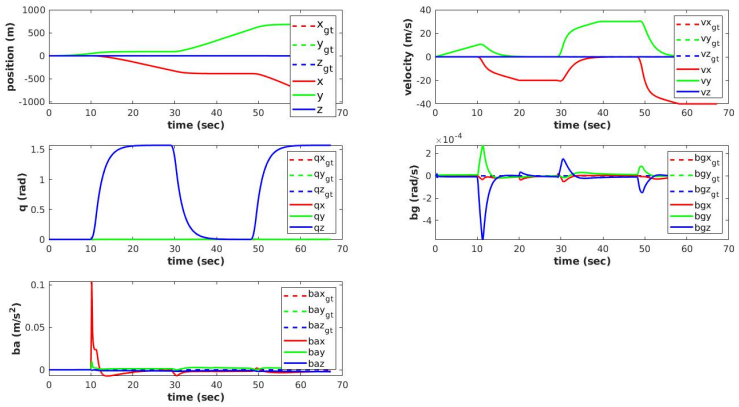
参数配置与上一题基本一致，只是添加了磁力计的误差 mag_std。另外，由于仿真程序输出的磁力计真值是在 b 系下，所以需要在读取仿真数据的时候将其转为 n 系下表示。

在上一题 ins 工程的基础上，添加磁力计的观测部分。参数配置中添加了 mag_noise 作为里程计噪声，use_mag 来选择是否启用磁力计观测。在 Correct 函数中添加磁力计观测，如下所示

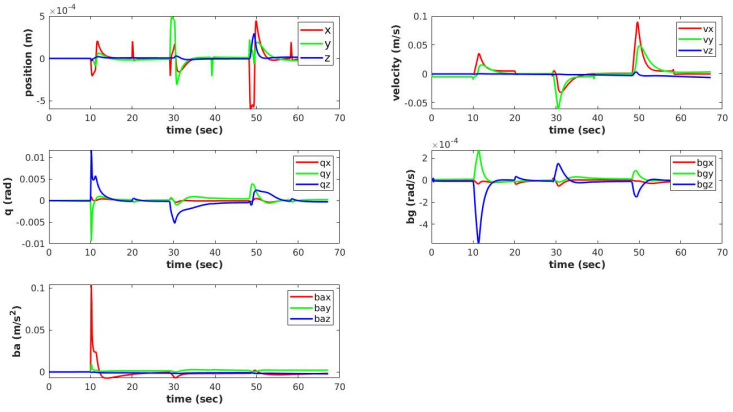
```
else if (use_mag)
{
    Eigen::Matrix3d cbn = current_pose.state.q.toRotationMatrix().transpose();
    Y.resize(6, 1);
    Y.block<3, 1>(0, 0) = current_pose.state.p - gnss_xyz;
    Y.block<3, 1>(3, 0) = cbn * current_mag.ref_mag - current_mag.mag;

    Gt = Eigen::Matrix<double, 6, 15>::Zero();
    Gt.block<3, 3>(0, 0) = Eigen::Matrix<double, 3, 3>::Identity();
    Eigen::Matrix3d bnx = Eigen::Matrix3d::Zero();
    bnx(0, 1) = - current_mag.ref_mag(2);
    bnx(0, 2) = current_mag.ref_mag(1);
    bnx(1, 0) = current_mag.ref_mag(2);
    bnx(1, 2) = -current_mag.ref_mag(0);
    bnx(2, 0) = -current_mag.ref_mag(1);
    bnx(2, 1) = current_mag.ref_mag(0);
    Gt.block<3, 3>(3, 6) = -cbn * bnx;

    Ct = Eigen::Matrix<double, 6, 6>::Identity();
    R = Eigen::Matrix<double, 6, 6>::Identity();
    R.block<3, 3>(0, 0) = Eigen::Matrix3d::Identity() * dp_noise * dp_noise;
    R.block<3, 3>(3, 3) = Eigen::Matrix3d::Identity() * mag_noise * mag_noise;
}
```

（3）仿真

仿真命令参见附件文件 imu_def16.csv，轨迹先往北走，再往西，再往北，再往西。从误差值的收敛情况可看出，所有状态量均收敛。



真实值与滤波值比较



误差值（滤波值减去真实值）