第二章作业:

Risingdragon

1. 设置 IMU 仿真代码中的不同参数，生成 allen 方差标定曲线：

使用的工具为 imu_utils,通过在网上查阅相关资料的。仿真程序的的参数设置在 vio_data_simulation-ros_version 的 param.h 文件中。设置如下

```
// noise
double gyro_bias_sigma = 0.00005;
double acc_bias_sigma = 0.0005;

//double gyro_bias_sigma = 1.0e-5;
//double acc_bias_sigma = 0.0001;

double gyro_noise_sigma = 0.015;    // rad/s
double acc_noise_sigma = 0.019;      //  m/(s^2)

double pixel_noise = 1;              // 1 pixel noise
```

标定后的 allen 方差曲线为：

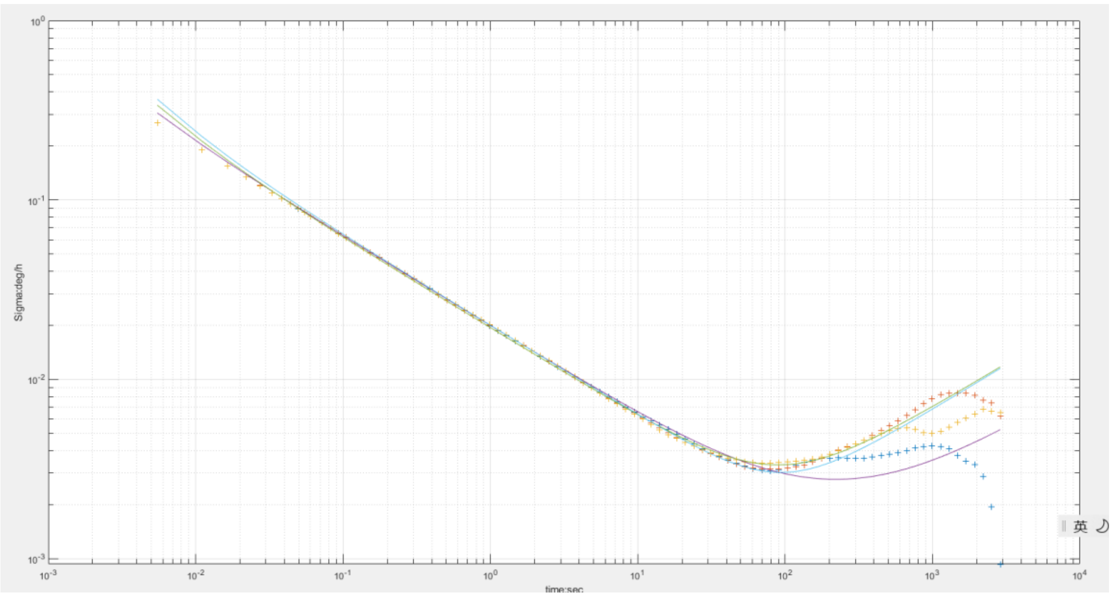

图 1 加速度的方差曲线

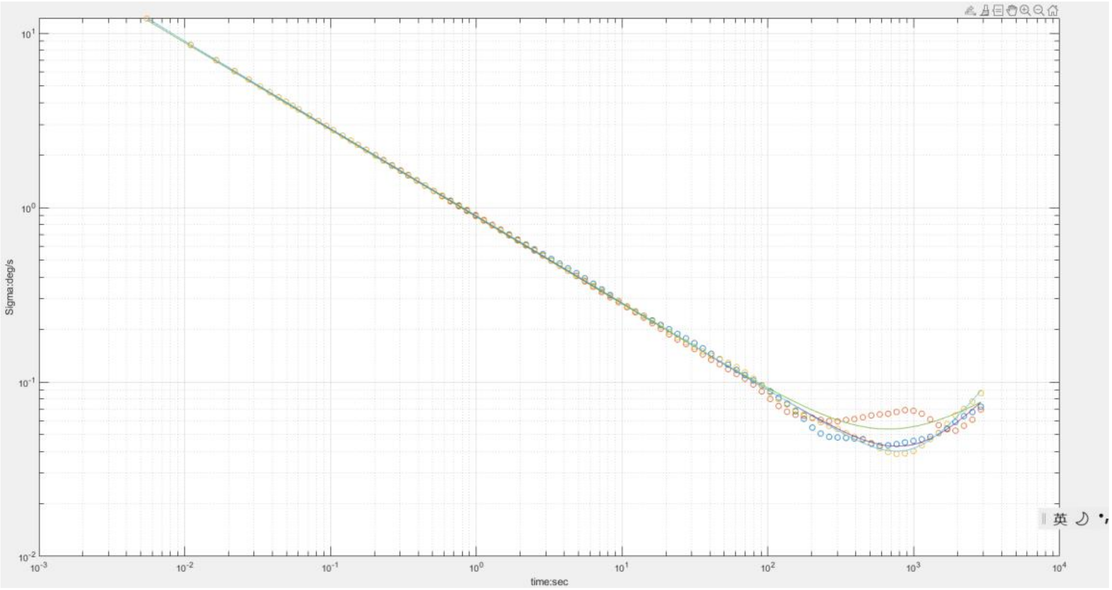

图 2 角速度的方差曲线

标定结果

表格 1 陀螺仪的标定结果

陀螺仪	设定值	标定值
白噪声	0.015	0.0148
偏置	0.00005	0.000056

表格 2 加速度计标定结果

加速度计	设定值	标定值
白噪声	0.019	0.0183
偏置	0.0005	0.0002

2.用欧拉积分替换中值积分

代码如下：

```
for (int i = 1; i < imudata.size(); ++i) {

        MotionData imupose = imudata[i];
        MotionData imu1 = imudata[i-1];     //上一个时刻的 IMU 信息

        Eigen::Quaterniond dq;
        Eigen::Vector3d dtheta_half = 0.5*(imupose.imu_gyro+imu1.imu_gyro) * dt /2.0;     //w = 1/2(w1+w2)
        dq.w() = 1;
        dq.x() = dtheta_half.x();
        dq.y() = dtheta_half.y();
        dq.z() = dtheta_half.z();
        dq.normalize();


        /// 中值积分
```

Eigen::Quaterniond Qwb1 = Qwb * dq;   //Qwb1 当前时刻的姿态四元数，Qwb 为上一时刻的

Eigen::Vector3d acc_w = 0.5*(Qwb1*imupose.imu_acc+Qwb*imu1.imu_acc)+gw; //中值法的公式，这里 gw 取符号选取根定义有关

Pwb = Pwb + Vw * dt + 0.5 * dt * dt * acc_w;

Vw = Vw + acc_w * dt;

Qwb=Qwb1;

结果：

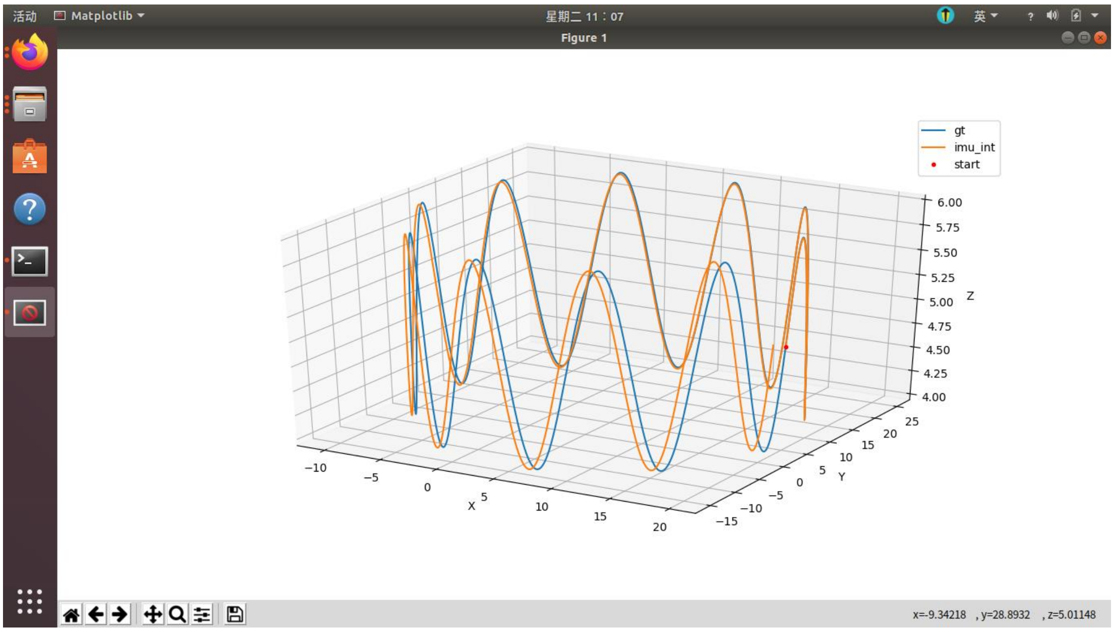

图 3 欧拉法

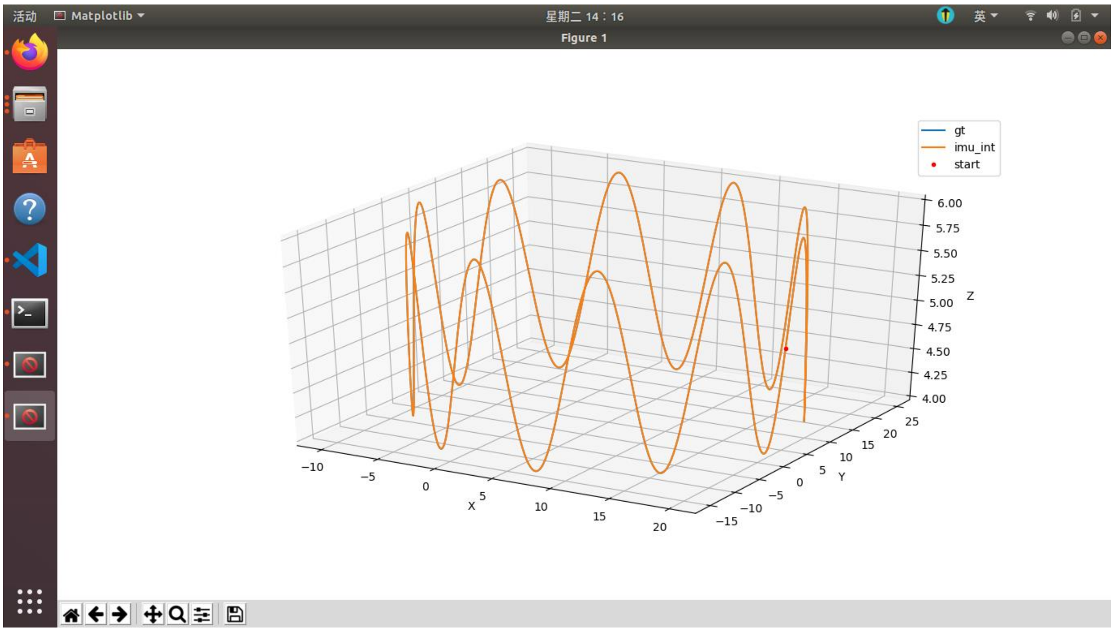

图 4 中值法

对比上述两个轨迹可得，中值法得到的轨迹更加贴近真实轨迹。

3.阅读从已有轨迹生成 imu 数据的论文，撰写总结推导：

总的来讲，b 样条法可以让我们得到所需要的轨迹的方程，得到轨迹方程之后进行微分操作得到需要的 IMU 数据。接下来进行详细的分析：

相机的位姿表示：

两帧之间的相机位姿可以用 4×4 的齐次变换矩阵 Q 表示为:

$$\boldsymbol{T}_{ba}=\begin{bmatrix}\boldsymbol{R}_{ba} & \boldsymbol{t}_{ba}\\ \boldsymbol{0}^T & 1\end{bmatrix},\ \boldsymbol{T}_{ba}\in\mathbb{SE}3,\ \boldsymbol{R}_{ba}\in\mathbb{SO}3$$

式中 $\boldsymbol{T}_{ba}$ 表示坐标系 a 到坐标系 b 之间的坐标变换矩阵，即 $x^b=\boldsymbol{T}_{ba}\boldsymbol{x}^a;\boldsymbol{R}_{ba}$ 为旋转变换矩阵， $\boldsymbol{t}_{ba}$ 为平移变换矢量，即 a 坐

标系远点在 b 坐标系下的坐标 $\boldsymbol{a}_b$ 。进一步地， a、b 两帧之间的角速度和线速度可以表示为 $\boldsymbol{\xi}^{\wedge}=\frac{1}{\Delta t}\log(\boldsymbol{T}_{ba}),\boldsymbol{\xi}^{\wedge}\in\ \mathbb{R}^{4\times 4}$ 。

$\mathbb{SE}3$ 上的曲线表征了刚体在空间中的运动轨迹，因此选取拟合曲线类型时应考虑如下几点: 1） 具有 C^2 连续性，即二阶导 与待拟合曲线一直;

2） 可通过控制点实现局部控制，从而便于在线运行；

3） 具有最小的力矩 (torque)Cubic B-Splines 是在 $\mathbb{R}^3$ 上进行轨迹表示的常用方法，但其在涉及旋转插值时的性 能会出现严重下降（例如无法保证 C^2 连续性)，因此本文使用改进的累积 Cubic B-Spline 方法来表示连续时间轨迹。

Cumulative cubic B-Splines 曲线

累积基函数

具有 k 个控制点的 $k-1$ 阶 B-Spline 曲残的标准基函数表示为:

$$\boldsymbol{p}(t)=\sum_{i=0}^{n}\boldsymbol{p}_iB_{i,k}(t)$$

式中 $p_i\in\mathbb{R}^N$ 为时间 $t_i,i=0,1,\cdots,n$ 处的控制点， $B_{i,k}(t)$ 称为 B-Spline 的基函数，其是一个标量值，表示不同控制点的权重且满足 De Boor-Cox 回归形式:

$$B_{i,0}(t)\quad :=\begin{cases}1, & t_i\le t\le t_{i+1}\\ 0, & \text{otherwise}\end{cases}$$

$$B_{i,k}(t):\quad =\frac{t-t_i}{t_{i+k}-t_i}B_{i,k-1}(t)+\frac{t_{i+(k+1)}-t}{t_{i+(k+1)}-t_{i+1}}B_{i+1,k-1}(t)$$

式（1）又可进一步写成细积形式：

$$\boldsymbol{p}(t)=\boldsymbol{p}_0\tilde{B}_{0,k}(t)+\sum_{i=1}^{n}(\boldsymbol{p}_i-\boldsymbol{p}_{i-1})\tilde{B}_{i,k}(t)$$

式中 $\tilde{B}_{i,k}(t)=\sum_{j=i}^{n}B_{j,k}(t)$ 为累积其函数。

对式(2)求取反对称并进行利用矩阵的指数映射和对数映射性质有：

$$\boldsymbol{T}_{w,s}(t)=\exp\left(\tilde{B}_{0,k}(t)\log\left(\boldsymbol{T}_{w,0}\right)\right)\prod_{i=1}^{n}\exp\left(\tilde{B}_{i,k}(t)\log\left(\boldsymbol{T}_{w,i-1}^{-1}\boldsymbol{T}_{w,i}\right)\right)$$

为了简化问题的建模和求解，实际应用中我们通常假设 B-Splilne 中的控制点是服从均匀分布的，即每个控制点之间的间隔固定为：$u(t)=\frac{t-t_i}{t_{i+1}-t_i},u(t)\in[0,1)$

时间微分：

$$\begin{aligned}
\dot{\mathbf{T}}_{w,s}(u) &= \mathbf{T}_{w,i-1}\left(\dot{\mathbf{A}}_0\mathbf{A}_1\mathbf{A}_2+\mathbf{A}_0\dot{\mathbf{A}}_1\mathbf{A}_2+\mathbf{A}_0\mathbf{A}_1\dot{\mathbf{A}}_2\right),\\
\ddot{\mathbf{T}}_{w,s}(u) &= \mathbf{T}_{w,i-1}\begin{pmatrix}\ddot{\mathbf{A}}_0\mathbf{A}_1\mathbf{A}_2+\mathbf{A}_0\ddot{\mathbf{A}}_1\mathbf{A}_2+\mathbf{A}_0\mathbf{A}_1\ddot{\mathbf{A}}_2+\\ 2\left(\dot{\mathbf{A}}_0\dot{\mathbf{A}}_1\mathbf{A}_2+\dot{\mathbf{A}}_0\mathbf{A}_1\dot{\mathbf{A}}_2+\mathbf{A}_0\dot{\mathbf{A}}_1\dot{\mathbf{A}}_2\right)\end{pmatrix},\\
\mathbf{A}_j &= \exp\left(\Omega_{i+j}\tilde{\mathbf{B}}(u)_j\right),\ \dot{\mathbf{A}}_j=\mathbf{A}_j\Omega_{i+j}\dot{\tilde{\mathbf{B}}}(u)_j,\\
\ddot{\mathbf{A}}_j &= \dot{\mathbf{A}}_j\Omega_{i+j}\dot{\tilde{\mathbf{B}}}(u)_j+\mathbf{A}_j\Omega_{i+j}\ddot{\tilde{\mathbf{B}}}(u)_j
\end{aligned}$$

IMU 量测模型：

根据上述微分方程，给出了世界坐标系 w 下用 B-Spline 表示的位姿方程，将其变换到传感器坐标系下并添加对应的噪声项即可获得 IMU 的模拟输出。

$$\begin{aligned}
\boldsymbol{\omega}_m &= \left(\boldsymbol{R}_{w,s}^T\boldsymbol{R}_{w,s}^T\right)^v+\boldsymbol{b}_\omega+\boldsymbol{n}_\omega\\
\boldsymbol{a}_m &= \boldsymbol{R}_{w,s}^T\left(\boldsymbol{t}_{w,s}+\boldsymbol{g}_w\right)+\boldsymbol{b}_d+\boldsymbol{n}_u
\end{aligned}$$