提交的作业包括三个部分：

（1）本文档，记录了代码运行过程和结果，以及对应的 PPT 知识点；论文的推导和理解

（2）手写笔记，主要是 B 样条曲线的一些公式说明，随机过程的学习没有记录下来

（3）代码包，原代码中添加了注释，修改了中值积分；标定工具只把当时的生成结果放进去了都放在了后缀为 data 的文件夹中，bag 和 mat 文件太大，压不到 100mb 以内，没放进来

作业并没有完成到最好，还有问题需要解决，问题已标注出来了

1、非 ros 版本生成带噪声的 imu 数据

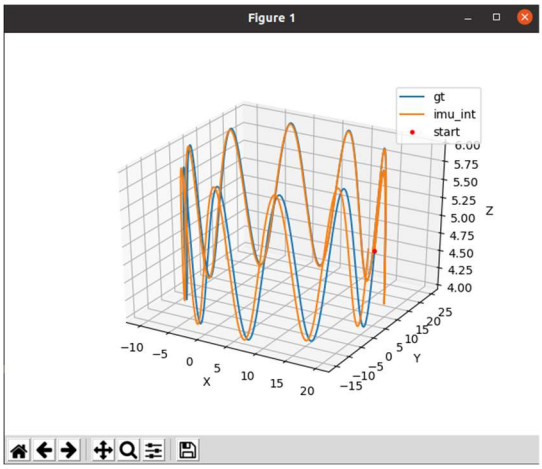
对代码每一步的理解在源码上已注释，主要是对 imu.cpp 的注释，第一遍阅读的理解在 ros 版本下写的，改中值积分时在非 ros 版本也做了一些注释，非 ros 版本的注释主要还是关于积分法的，截图中已经体现了，我自己也分不清原版有哪些注释了。

欧拉积分法代码（原代码中已含）

```
for (int i = 1; i < imudata.size(); ++i) {
    //以下是欧拉积分代码
    //k时刻的信息
    MotionData imupose = imudata[i];
    // delta_q = [1 , 1/2 * thetax , 1/2 * theta_y, 1/2 * theta_z]
    Eigen::Quaterniond dq;
    Eigen::Vector3d dtheta_half = imupose.imu_gyro * dt / 2.0;
    dq.w() = 1;
    dq.x() = dtheta_half.x();
    dq.y() = dtheta_half.y();
    dq.z() = dtheta_half.z();
    dq.normalize();

    /// imu 动力学模型 欧拉积分
    Eigen::Vector3d acc_w = Qwb * (imupose.imu_acc) + gw;  // aw = Rwb * ( acc_body - acc_bias ) + gw
    Qwb = Qwb * dq;
    Pwb = Pwb + Vw * dt + 0.5 * dt * dt * acc_w;
    Vw = Vw + acc_w * dt;
    //以上是欧拉积分代码
```

生成数据并画图结果

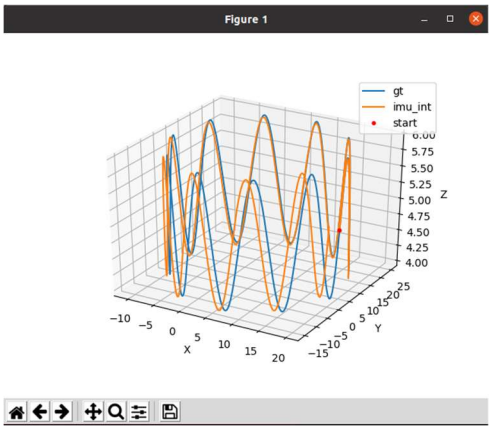


中值积分法代码（自己写的）

一开始多取的 imudata 中的 i+1 索引值，明显曲线偏离更大了

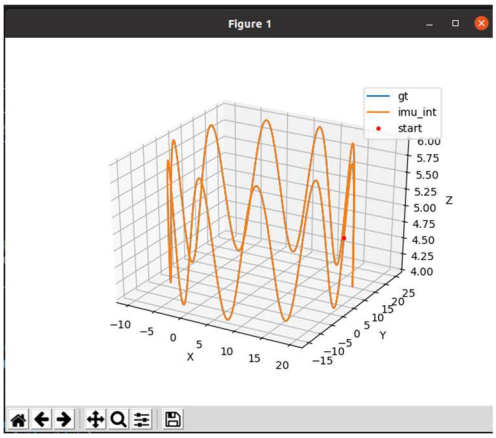
```
    /// 中值积分
    //k时刻的信息
    MotionData imupose = imudata[i];
    //k+1时刻的信息
    MotionData imupose_k1 = imudata[i+1];
    //利用中值法计算k+1时刻的旋转四元数扰动量
    Eigen::Quaterniond dq_midpoint;
    Eigen::Vector3d dtheta_half = ( imupose.imu_gyro + imupose_k1.imu_gyro ) / 2.0 * dt /2.0;
    dq_midpoint.w() = 1;
    dq_midpoint.x() = dtheta_half.x();
    dq_midpoint.y() = dtheta_half.y();
    dq_midpoint.z() = dtheta_half.z();
    dq_midpoint.normalize();
    //利用中值法求世界坐标系下IMU的加速度，好像没有算bias随机游走？？？
    Eigen::Vector3d acc_w = (1/2.0) *((Qwb * (imupose.imu_acc) + gw) + (Qwb * dq_midpoint * (imupose_k1.imu_acc) + gw));  // aw = Rwb * ( acc_body - acc_bias ) + gw
    Qwb = Qwb * dq_midpoint;
    Pwb = Pwb + Vw * dt + 0.5 * dt * dt * acc_w;
    Vw = Vw + acc_w * dt;
    //以上是中值积分代码
```

生成数据并画图结果



索引值改为 i-1 后：

```
//中值积分
//k时刻的信息
MotionData imupose = imudata[i];
//k+1时刻的信息
MotionData imupose_k1 = imudata[i-1];
//利用中值法计算k+1时刻的旋转四元数扰动量
Eigen::Quaterniond dq_midpoint;
Eigen::Vector3d dtheta_half = ( imupose.imu_gyro + imupose_k1.imu_gyro ) / 2.0 * dt /2.0;
dq_midpoint.w() = 1;
dq_midpoint.x() = dtheta_half.x();
dq_midpoint.y() = dtheta_half.y();
dq_midpoint.z() = dtheta_half.z();
dq_midpoint.normalize();
//利用中值法求世界坐标系下IMU的加速度，好像没有算bias随机游走？？？
Eigen::Vector3d acc_w = (1/2.0) *((Qwb * (imupose.imu_acc) + gw) + (Qwb * dq_midpoint * (imupose_k1.imu_acc) + gw));  // aw = Rwb * ( acc_body - acc_bias ) + gw
Qwb = Qwb * dq_midpoint;
Pwb = Pwb + Vw * dt + 0.5 * dt * dt * acc_w;
Vw = Vw + acc_w * dt;
//以上是中值积分代码
```



关于这个问题的思考，自己写的中值法思路没有问题，但是选择前一个点还是后一个点搞错了，而且目前还没有搞清楚，PPT 的意思是欧拉法 k+1 时刻的点用 k 时刻算，k+1 时刻的点用 k 和 k+1 时刻算，那修改的话不应该是添加上原来取的 i 和 i+1 的信息吗？就是还没有搞清楚，哪个代表 k 时刻，哪个代表 k+1 时刻，现在处理的又是哪个时刻，后续再思考。

2、allan 方差标定曲线

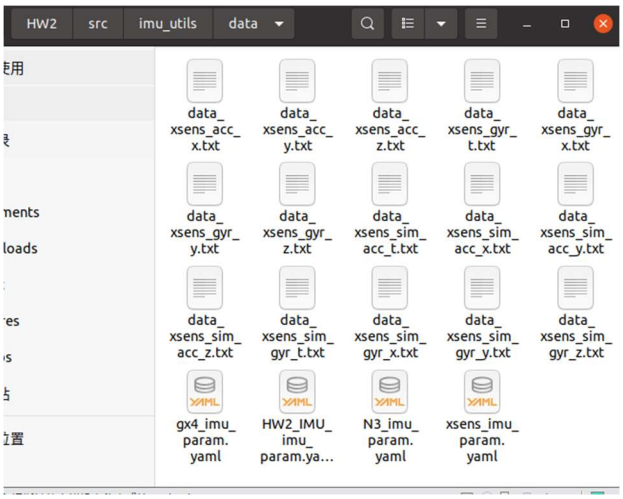
首先在 ros 中利用 vio_data_simulation 功能包下的 vio_data_simulation_node 节点生成 imu 仿真数据，名称为 imu.bag，保存到了主目录下。

Imu_utils:

按照 imu_utils 的教程运行 imu_utils 工具

```
<launch>
    <node pkg="imu_utils" type="imu_an" name="imu_an" output="screen">
        <param name="imu_topic" type="string" value= "imu"/>
        <param name="imu_name" type="string" value= "HW2_IMU"/>
        <param name="data_save_path" type="string" value= "$(find imu_utils)/data/"/>
        <param name="max_time_min" type="int" value= "120"/>
        <param name="max_cluster" type="int" value= "100"/>
        <remap from="/imu_an/imu" to="/imu"/>
    </node>
</launch>
```

Launch 文件中运行 imu_utils 功能包下的 imu_an 节点,IMU 取名为 HW2_IMU, 数据存在 data 文件夹下，同时将 imu_an 节点订阅的话题名称由/imu_an/imu 改为/imu，因为我将在主目录下回放存有 imu 仿真数据的 bag 文件，回放时的话题是/imu。
利用 rosbag 回放 imu 仿真数据，用 imu_utils 接收产生结果，结果存在 data 文件夹中，生成了一些 txt 文件，是一些具体的分析数据，还有一个 yaml 文件：



图中的 HW2_IMU_imu_param.yaml 即是本次实验的结果。
内容如下：

```
%YAML:1.0
---
type: IMU
name: HW2_IMU
Gyr:
   unit: " rad/s"
   avg-axis:
      gyr_n: 2.0959891122675942e-01
      gyr_w: 7.5827540553341087e-04
   x-axis:
      gyr_n: 2.0265292680555058e-01
      gyr_w: 5.6485292781995175e-04
   y-axis:
      gyr_n: 2.1428089384045662e-01
      gyr_w: 8.9229416529235171e-04
   z-axis:
      gyr_n: 2.1186291303427104e-01
      gyr_w: 8.1767912348792917e-04
Acc:
   unit: " m/s^2"
   avg-axis:
      acc_n: 2.6717534497709766e-01
      acc_w: 3.5263417801519978e-03
   x-axis:
      acc_n: 2.6649342503605244e-01
      acc_w: 3.4208921689030838e-03
   y-axis:
      acc_n: 2.6629243060098096e-01
      acc_w: 3.3423094615410492e-03
   z-axis:
      acc_n: 2.6874017929425947e-01
      acc_w: 3.8158237100118608e-03
```

意义如下：（单位对不上？离散时间下高斯白噪声的单位是原数据单位除以 sqrt(s)，但是表中除以的是 sqrt(Hz)）

IMU Noise Values

Parameter	YAML element	Symbol	Units
Gyroscope "white noise"	`gyr_n`	σ_g	$\frac{rad}{s}\frac{1}{\sqrt{Hz}}$
Accelerometer "white noise"	`acc_n`	σ_a	$\frac{m}{s^2}\frac{1}{\sqrt{Hz}}$
Gyroscope "bias Instability"	`gyr_w`	σ_{bg}	$\frac{rad}{s}\sqrt{Hz}$
Accelerometer "bias Instability"	`acc_w`	σ_{ba}	$\frac{m}{s^2}\sqrt{Hz}$

- White noise is at tau=1;
- Bias Instability is around the minimum;

(according to technical report: `Allan Variance: Noise Analysis for Gyroscopes`)

将 avg-axis 的结果转换到连续时间下，与我们生成的 imu 仿真数据时的设定作对比：

高斯白噪声：(200Hz 采样频率，采样时间是 1/200s)

陀螺仪：gyro.noise.sigma=2.0959891122675942e-01*sqrt(1/200)=0.0148

加速度计：acc.noise.sigma=2.6717534497709766e-01*sqrt(1/200)=0.0189

Bias 随机游走：

陀螺仪：gyro.bias.sigma=7.5827540553341087e-04*sqrt(200)=0.0107

加速度计：acc.bias.sigma=3.5263417801519978e-03*sqrt(200)=0.0499

和下面的设定对比发现 Bias 测的不准

生成仿真数据时对参数的设定：

```
public:

    Param();

    // time
    int imu_frequency = 200;
    int cam_frequency = 30;
    double imu_timestep = 1./imu_frequency;
    double cam_timestep = 1./cam_frequency;
    double t_start = 0;
    double t_end = 3600 * 4;  // 4 hours for allan
```

Imu 频率为 200hz，所以传感器采样时间就是 1/200 s。

```
    // 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 * 1/sqrt(hz)
    double acc_noise_sigma = 0.019;      //  m/(s^2) * 1/sqrt(hz)
```

vio_data_simulation-ros_version/src/param.h 代码中，有关于陀螺仪和加速度计的白噪声和随机游走的参数设置，bias 是随机游走，noise 是白噪声，gryo 是陀螺仪，acc 是加速度计。

（回放速率对结果有什么影响？）

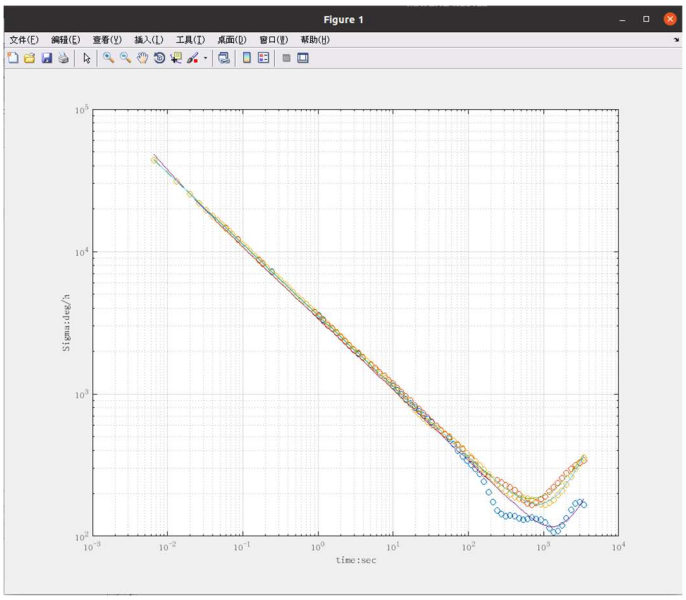
500 倍速率回放结果：

```yaml
%YAML:1.0
---
type: IMU
name: HW2_IMU
Gyr:
   unit: " rad/s"
   avg-axis:
      gyr_n: 2.1136369846263200e-01
      gyr_w: 8.0880565839794251e-04
   x-axis:
      gyr_n: 2.1083601119494186e-01
      gyr_w: 6.5728367877276482e-04
   y-axis:
      gyr_n: 2.1405325066661832e-01
      gyr_w: 9.1426644429497229e-04
   z-axis:
      gyr_n: 2.0920183352633587e-01
      gyr_w: 8.5486685212609031e-04
Acc:
   unit: " m/s^2"
   avg-axis:
      acc_n: 2.6686303921772841e-01
      acc_w: 3.4908611387581179e-03
   x-axis:
      acc_n: 2.6661100915322622e-01
      acc_w: 3.5635582180175702e-03
   y-axis:
      acc_n: 2.6846254217884347e-01
      acc_w: 3.3028741344246905e-03
   z-axis:
      acc_n: 2.6551556632111545e-01
      acc_w: 3.6061510638320938e-03
```

1000 倍速率回放结果:

```yaml
%YAML:1.0
---
type: IMU
name: HW2_IMU
Gyr:
   unit: " rad/s"
   avg-axis:
      gyr_n: 2.0932374894125372e-01
      gyr_w: 8.0950901837144134e-04
   x-axis:
      gyr_n: 2.0426190127203761e-01
      gyr_w: 4.8670647537517873e-04
   y-axis:
      gyr_n: 2.1381478687115157e-01
      gyr_w: 9.9728082061462716e-04
   z-axis:
      gyr_n: 2.0989455868057194e-01
      gyr_w: 9.4453975912451779e-04
Acc:
   unit: " m/s^2"
   avg-axis:
      acc_n: 2.6726709286022854e-01
      acc_w: 3.8526515971040042e-03
   x-axis:
      acc_n: 2.6581044209750843e-01
      acc_w: 3.8323091197075285e-03
   y-axis:
      acc_n: 2.6532302875673935e-01
      acc_w: 3.7594415361055281e-03
   z-axis:
      acc_n: 2.7066780772643778e-01
      acc_w: 3.9662041354989568e-03
```

标准差差的似乎不多，好像没啥影响，深层原因还不清晰，对 rosbag 回放机制和 imu_utils 标定机制理解不够。



生成的

Kalibr_allan:

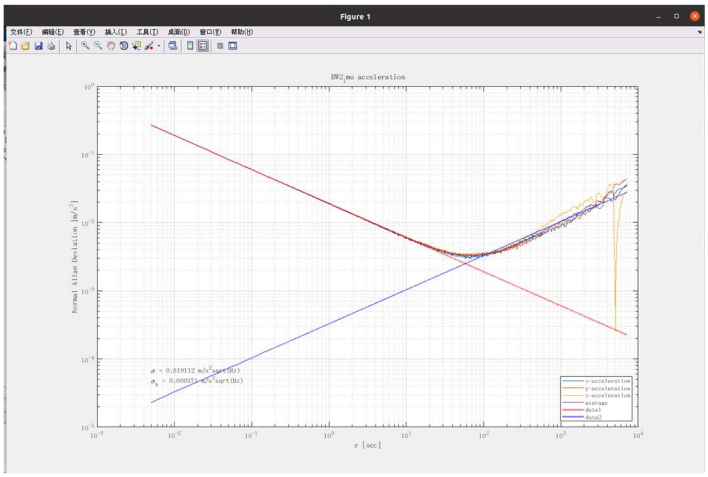
```
>> SCRIPT_allan_matparallel
opening the mat file.
loading timeseries.
imu frequency of 200.00.
sample period of 0.00500.
calculating allan deviation.
时间已过 885.890665 秒。
```

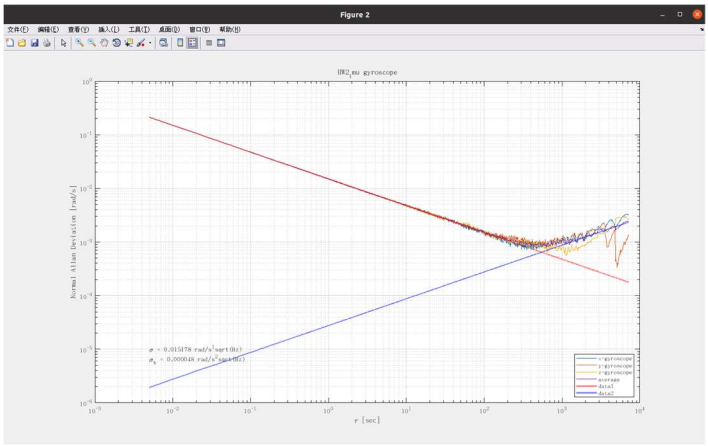
虚拟机运行 10 分钟……

```
>> SCRIPT_process_results
=> opening the mat file.
=> plotting accelerometer.
警告: MATLAB 已通过改用 OpenGL 软件禁用了某些高级的图形渲染功能。欲了解有关详细信息，请点击此处。
tau = 1.00 | tauid1 = 1089
h_fit1 slope = -0.5000 | y-intercept = -3.9574
h_fit2 slope = 0.5000 | y-intercept = -8.0151
tau = 2.99 | tauid2 = 1201
=> plotting gyroscope.
tau = 1.00 | tauid1 = 1089
h_fit1 slope = -0.5000 | y-intercept = -4.1879
h_fit2 slope = 0.5000 | y-intercept = -10.4866
tau = 2.99 | tauid2 = 1201
=> final results
accelerometer_noise_density = 0.01911224
accelerometer_random_walk   = 0.00057140
gyroscope_noise_density     = 0.01517814
gyroscope_random_walk       = 0.00004826
```

运行结果已经显示出了算好的连续时间的 noise 和 random_walk，不需要自己转换，下图中也有标注。

可见随机游走测量的更加准确了，都很接近生成 imu 仿真数据的设定值





3、论文总结

Abstract

本文描述了用于视觉+惯导融合同时定位建图和校准的通用连续时间框架

1 Introduction

2 Continuous-time representation 连续时间描述

使用连续时间描述可以显著减少参数，视觉和惯性器件可以在最小二乘最小化范围内统一（统一的意思不是很清楚，大概是如果利用连续时间描述，视觉和惯性器件可以定义出最小二乘法的问题，并都能达到最优解吧）

由于选择使用 Cayley-Gibbs Rodrigues 公式参数化姿态（表示位姿），然后在该空间内进行插值，连续时间表示法不适合使用样条曲线进行精确表示。（不明白为什么，总之不能用样条曲线精确表示）

连续时间表示法有三个缺点：1、180°的时候有奇异性 2、该空间中的插值不会反映旋转组中的最小距离 3、它不能很好地逼近转矩最小轨迹（具体原因均不清楚）

相反，我们选择李群 SE3 的李代数 se3，以便在刚体平移和旋转的流形空间（十四讲中提过，实际知识不懂）中实现平滑的轨迹（大概就是用李群李代数表述位姿会更好，我们一直用的也都是李群李代数，用来求导啥的），然后这种参数化方式可以避免奇异性，还可以给出最小扭矩轨迹的解析近似。（弥补前面方法的缺点）

2.1Camera pose transformations

先介绍 SE3 代表的转移矩阵

$$\mathbf{T}_{b,a} = \begin{bmatrix} \mathbf{R}_{b,a} & \mathbf{a}_b \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathbf{T}_{b,a} \in \mathbb{SE}3, \quad \mathbf{R}_{b,a} \in \mathbb{SO}3,$$

Δt时间由a位姿一恒定的角速度和线速度到b位姿，速度可以表示为如下矩阵，log 是矩阵对数映射（不明白对数映射是如何变成矩阵的），但是论文中说对数映射及其逆映射（指数映射）可以以封闭形式计算，并给了参考文献，所以应该是有具体方法，不深究了

$$\boldsymbol{\Omega} = \frac{1}{\Delta t} \log\left(\mathbf{T}_{b,a}\right), \boldsymbol{\Omega} \in \mathbb{R}^{4\times4},$$

2.2C^2-continuous curves in SE3

方法的核心是连续的轨迹表示。

我们选择了一种公式，该公式提供：（i）局部控制，允许系统在线和批量运行（估计是可以批处理，或者有递推的形式，所以可以在线及时更新）；（ii）C^2 连续性（C^2 连续好像是二次导数连续的意思，G^2 指曲面连续？有待深究），使我们能够预测 IMU 测量值；以及（iii）最小扭矩轨迹的良好近似。三次 B 样条曲线（有关 B 样条曲线在手写的笔记中，已经搞清楚了，是数值分析中一种通过给定控制点拟合出近似曲线的方法，基于贝塞尔曲线，但是弥补了贝塞尔曲线的不足，有很好的应用性质，灵活度更高）是 R3（三维实空间）中众所周知的轨迹表示，但在处理 3D 旋转（如 SO3 中的插值）时不太容易应用。例如，C^2 连续性不一定保持。我们选择使用李代数形成的累积基函数对连续轨迹进行参数化，与[8]中提出的等效（大概就是三次 B 样条曲线不好应用在旋转矩阵中，所以把 B 样条曲线中的基函数替换成论文提出的累计基函数，累计基函数是用李代数表示的，用李代数好算旋转矩阵的东西，可以保证旋转矩阵的性质不变）。[16]中，在计算机动画的背景下，首次提出了使用累积 B 样条基函数进行四元数插值。该表示不仅是 C^2 连续的，而且还提供了一个非常简单的二阶导数公式。（对上面方法的补充说明，也就是用累积基函数不仅保证 C^2 连续，也不知道是啥东西保证 C^2 连续，估计是一系列旋转矩阵代表的轨迹？，然后参考论文还给了二阶导数的公式，因为 C^2 连续就是二阶导数连续）

2.2.1Representing B-Splines with cumulative basis functions

标准 k 阶 B 样条基函数，次数为 k-1，p_i是控制点

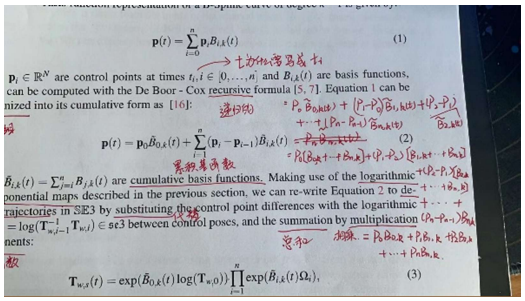
$$\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,k}(t)$$

De Boor-Cox 递推式手写笔记中有，这里写成了累积的形式：

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

累积基函数，由标准基函数求和从第 i 个标准基函数开始累加，最终的 B 样条曲线上的点表示为分别从第 0，1，……n 个标准基函数开始到第 n 个标准基函数的求和，最终再加到一起

推导过程：



用对数和指数映射重写累积基函数的公式：

$$\Omega_i = \log(\mathbf{T}_{w,i-1}^{-1}\mathbf{T}_{w,i}) \in \mathfrak{se}3$$

i 姿态到 i-1 姿态的 se3 李代数，这里和前面的Ω意义不一样了，前面对数映射后除以时间表示速度的是 R4X4，这里变成了正常的李代数，不明白。

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

用姿态代替了原来的控制点，用指数相乘替代了原来的求和

where $\mathbf{T}_{w,s}(t) \in \mathbb{SE}3$ is the pose along the spline at time t, and $\mathbf{T}_{w,i} \in \mathbb{SE}3$ are the control poses. We use the subscript w to emphasize that the pose at time t and control poses are given in the world coordinate frame.

$T_{w,s}(t)$代表沿着 B 样条的各个姿态，参考世界坐标系，$T_{w,i}$代表控制姿态

2.2.2Cumulative cubic B-Splines 累积三次 B 样条

论文里通过对时间变量的代换，把 t 换成了 u，我自己学习的标准 B 样条本身就是 u，不是时间，但是没有学习到 u 代表的意义是什么，这里的变量代换原因也没有搞清楚。

总之，变量代换后，导数的表示和原来的 pose 表示更简洁了：

$$\tilde{\mathbf{B}}(u) = \mathbf{C}\begin{bmatrix}1\\u\\u^2\\u^3\end{bmatrix}, \quad \dot{\tilde{\mathbf{B}}}(u) = \frac{1}{\Delta t}\mathbf{C}\begin{bmatrix}0\\1\\2u\\3u^2\end{bmatrix}, \quad \ddot{\tilde{\mathbf{B}}}(u) = \frac{1}{\Delta t^2}\mathbf{C}\begin{bmatrix}0\\0\\2\\6u\end{bmatrix}, \quad \mathbf{C} = \frac{1}{6}\begin{bmatrix}6&0&0&0\\5&3&-3&1\\1&3&3&-2\\0&0&0&1\end{bmatrix}$$

The pose in the spline trajectory can now be defined as:

$$\mathbf{T}_{w,s}(u) = \mathbf{T}_{w,i-1}\prod_{j=1}^{3}\exp\left(\tilde{\mathbf{B}}(u)_j\Omega_{i+j}\right), \tag{4}$$

进一步简化表示姿态的一阶和二阶导数：

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

3Generative model of visual-inertial data

3.1Parameterization

用第一帧的逆深度参数化地标，好处是可以简单处理无穷远处的点，简化单目视觉的特征初始化（不懂）

$$\mathbf{p}_b = \mathcal{W}(\mathbf{p}_a; \mathbf{T}_{b,a}, \rho) = \pi\left([\mathbf{K}_b \,|\, \mathbf{0}]\, \mathbf{T}_{b,a} \begin{bmatrix} \mathbf{K}_a^{-1} \begin{bmatrix} \mathbf{p}_a \\ 1 \end{bmatrix}; \rho \end{bmatrix}\right), \tag{7}$$

where $\pi(\mathbf{P}) = \frac{1}{\mathbf{P}_2}[\mathbf{P}_0, \mathbf{P}_1]^\intercal$ is the homogeneous projection function and $\mathbf{K}_a$, $\mathbf{K}_b \in \mathbb{R}^{3\times3}$ are the camera intrinsics for frames a and b respectively.

逆深度点的像素坐标可以在两个单目相机的像素坐标系转化，K 是相机内参，逆深度ρ不知道怎么加进去的，因为是非负实数，不是矩阵，维数对不上了，其他的维数能对上，不过P_2不知道代表什么

累积 B 样条曲线参数化允许我们计算分析时间导数，如第 2.2.2 节所示。这使得我们可以简单地综合加速计和陀螺仪的测量结果，然后我们可以使用它们来形成观测测量结果的误差。

$$\begin{aligned} \text{Gyro}(u) &= \mathbf{R}_{w,s}^\intercal(u) \cdot \dot{\mathbf{R}}_{w,s}(u) + \text{bias}, \\ \text{Accel}(u) &= \mathbf{R}_{w,s}^\intercal(u) \cdot (\ddot{\mathbf{s}}_w(u) + g_w) + \text{bias}, \end{aligned}$$

论文说 R 和 s 是转移矩阵 T 的适当的子矩阵，不了解更具体的内容，估计就是 T 的旋转和平移部分有关，g 是重力加速度，也是三维向量，维数都能对上，其实也不知道为什么这样算就是观测误差

3.2Minimization

给定我们的视觉和惯性数据生成模型，我们可以通过最小化由测量值与预测值的差异形成的目标函数来批量或在窗口内求解样条曲线和相机参数。通过使用连续时间公式，可以统一处理重投影误差（重投影误差忘记了，好像是和最小二乘有关？或者和多视图几何有关？记不清了）和惯性误差（不知道误差的具体表现形式），并通过根据设备规范或校准计算出的它们各自的信息矩阵Σ进行加权。当我们收到新的视觉惯性测量值时，我们反复尝试寻找$argmin_\theta E(\theta)$:

$$\begin{aligned} E(\theta) = &\sum_{\hat{\mathbf{p}}_m} \left(\hat{\mathbf{p}}_m - \mathcal{W}(\mathbf{p}_r; \mathbf{T}_{c,s}\mathbf{T}_{w,s}(u_m)^{-1}\mathbf{T}_{w,s}(u_r)\mathbf{T}_{s,c}, \rho)\right)^2_{\Sigma_p} + \\ &\sum_{\hat{\omega}_m} \left(\hat{\omega}_m - \text{Gyro}(u_m)\right)^2_{\Sigma_\omega} + \sum_{\hat{\mathbf{a}}_m} \left(\hat{\mathbf{a}}_m - \text{Accel}(u_m)\right)^2_{\Sigma_\mathbf{a}}, \end{aligned}$$

转化成了优化问题，并且给出了损失函数，还是叫代价函数误差函数？最小化就行

然后是对公式符号的解释，我不是很清楚，论文说通过迭代非线性最小二乘法实现这一点，需要线性化$E(\theta)$，找到这个近似函数的最小值，更新θ并重复。我们通过 se3 中相应的李代数进一步参数化姿势变换更新（给了参考论文）。

使用灵活的最小二乘解算器——Ceres 库，以便将此目标最小化。

后面具体应用到卷帘式照相机没看了。