ex.1

使用第二章的VIO仿真数据集输入VINS课程代码，并使用evo评估与ground truth的对比效果。

准备工作

1．设定仿真数据集的IMU采集频率为100hz，相机的采集频率为10hz。这个跟VINS_Course里面

发布IMU和相机线程的频率保持一致。

2．IMU和相机之间的外参设置为准确的，不需要标定，如下：

```
# Extrinsic parameter between IMU and Camera.
estimate_extrinsic: 0   # 0  Have an accurate extrinsic parameters. We will trust the following imu^R_cam, imu^T_cam, don't change it.
                        # 1  Have an initial guess about extrinsic parameters. We will optimize around your initial guess.
                        # 2  Don't know anything about extrinsic parameters. You don't need to give R,T. We will try to calibrate it. Do
#If you choose 0 or 1, you should write down the following matrix.
#Rotation from camera frame to imu frame, imu^R_cam
extrinsicRotation: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [0., 0., -1.,
          -1., 0., 0.,
          0., 1., 0.]
#Translation from camera frame to imu frame, imu^T_cam
extrinsicTranslation: !!opencv-matrix
   rows: 3
   cols: 1
   dt: d
   data: [0.05, 0.04, 0.03]
```

3．仿真数据集IMU噪声参数设置如下，注意这里设定的是连续时间下噪声标准差：

```
// time
int imu_frequency = 100; //200
int cam_frequency = 10; //30
double imu_timestep = 1./imu_frequency;
double cam_timestep = 1./cam_frequency;
double t_start = 0.;
double t_end = 20;  //  20 s

// noise
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)
```

而VINS代码中参数是噪声标准差，但是是离散的，所以IMU的高斯噪声需要再除以$\sqrt{\triangle t}$（或者乘以$\sqrt{100}$）,随机游走需要再乘以$\sqrt{\triangle t}$（或者除以$\sqrt{100}$）：

```
#仿真数据集里面的参数就是标准差，vins代码里面用的是方差，所以这里只需要转成离散时间
#imu parameters       The more accurate parameters you provide, the better performance
acc_n: 0.19  #0.019          # accelerometer measurement noise standard deviation. #0.2   0.04
gyr_n: 0.15 #0.015           # gyroscope measurement noise standard deviation.     #0.05  0.004
acc_w: 1.0e-5  #0.0001          # accelerometer bias random work noise standard deviation.  #0.02
gyr_w: 1.0e-6  #1.0e-5       # gyroscope bias random work noise standard deviation.     #4.0e-5
g_norm: 9.81007     # gravity magnitude
```

４．数据集时长总共20ｓ,因此共有2000帧IMU数据，200帧图像观测数据，每帧图像观测到36个特征点，存为归一化图像坐标，这里需要修改MAX_CNT为36：

```
#feature traker paprameters
max_cnt: 36            # max feature number in feature tracking
min_dist: 30            # min distance between two features
freq: 10                # frequence (Hz) of publish tracking result. At least 10Hz for good estimation. If set 0, the frequence will be sa
F_threshold: 1.0        # ransac threshold (pixel)
show_track: 1           # publish tracking image as topic
equalize: 1             # if image is too dark or light, trun on equalize to find enough features
fisheye: 0              # if using fisheye, trun on it. A circle mask will be loaded to remove edge noisy points
```

５．VINS代码里面主要修改了两处，一处是原来读取图片发布图像的线程改成了发布每帧图像的特征点：

```
void PubImageData()
{
#ifdef USE_SIMULATION // 200帧图像，每幅图像36个特征点
    string header_txt = "all_points_";
    string tail_txt = ".txt";
    for (unsigned int i = 0; i < 200; i++)
    {
        string id = to_string(i);
        string sFeatures_data_file = sConfig_path + header_txt + id + tail_txt;
        //cout << "1 PubFeaturesData start sFeatures_data_file: " << sFeatures_data_file << endl;
        ifstream fsFeatures;
        fsFeatures.open(sFeatures_data_file.c_str());
        if (!fsFeatures.is_open())
        {
            cerr << "Failed to open features file! " << sFeatures_data_file << endl;
            return;
        }

        std::string sFeatures_line;
        double dStampNSec = 0.0;
        vector<cv::Point2f> vFeatures;
        cv::Point2f feature;
        while (std::getline(fsFeatures, sFeatures_line) && !sFeatures_line.empty()) // read features data
        {
            std::istringstream ssFeaturesData(sFeatures_line);
            ssFeaturesData >> dStampNSec >> feature.x >> feature.y;
            //cout << "feature t: " << fixed << dStampNSec << " x: " << feature.x << " y: " << feature.y << endl;
            vFeatures.emplace_back(feature);
        }
        pSystem->PubImageData(dStampNSec, vFeatures);
        usleep(50000*nDelayTimes); //100ms 10hz
        fsFeatures.close();
    }

#else
```

第二处是trackerData[0].readImage函数：

```cpp
#ifdef USE_SIMULATION
void FeatureTracker::readImage(const std::vector<cv::Point2f>& features, double _cur_time)
{
    TicToc t_r;
    cur_time = _cur_time;
    forw_pts.clear();

    if (cur_pts.size() > 0) //已经有了上一帧的特征点
    {
        TicToc t_o;
        vector<uchar> status;
        // 当前帧的特征点
        forw_pts = features;

        for (int i = 0; i < int(forw_pts.size()); i++)
            status.emplace_back(1);
        // 剔除没有跟踪上的点,status[i]为1表示全部跟踪上了
        reduceVector(prev_pts, status);
        reduceVector(cur_pts, status);
        reduceVector(forw_pts, status);
        reduceVector(ids, status);
        reduceVector(cur_un_pts, status);
        reduceVector(track_cnt, status);
    }
    // 当前帧图像上跟踪到了，对应的图像id也加1，第一帧track_cnt[i]全是1
    for (auto &n : track_cnt)
        n++;

    if (PUB_THIS_FRAME)
    {
        // 第一帧forw_pts为空
        TicToc t_t;
        int n_max_cnt = MAX_CNT - static_cast<int>(forw_pts.size());
        if (n_max_cnt > 0)
        {
            // 获取特征点
            n_pts = features;
        }
        else
            n_pts.clear();   // 以后每一帧forw_pts都有36个观测

        // 将n_pts加入， id 初始化为-1, forw_pts,track_cnt
        TicToc t_a;
        addPoints();
        //ROS_DEBUG("selectFeature costs: %fms", t_a.toc());
    }
    prev_pts = cur_pts;
    prev_un_pts = cur_un_pts; //保存上一帧提取的好的特征点
    cur_pts = forw_pts; // 保存当前帧跟踪到的点
    undistortedPoints();
    prev_time = cur_time;
}
#else
```

测试结果

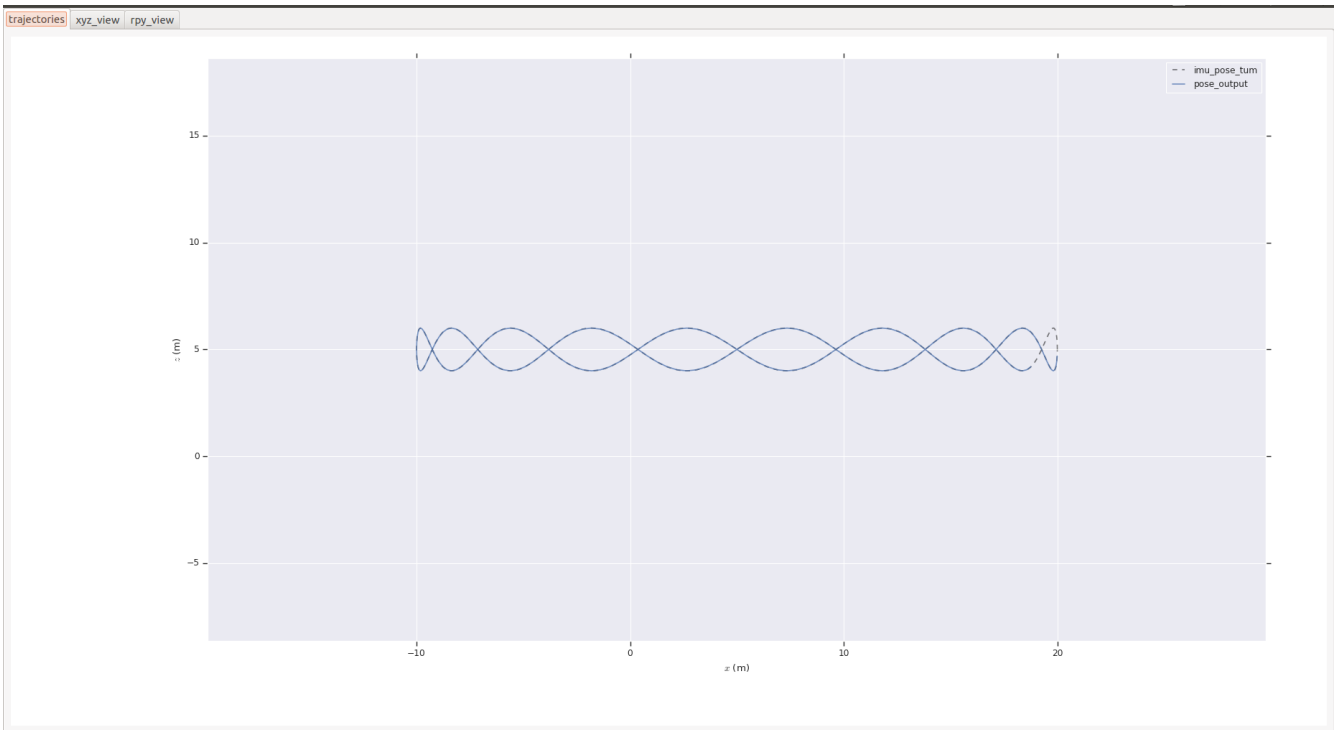
不含噪声的IMU数据保存在imu_data.txt,含噪声的数据保存在imu_noise_data.txt:

```
void PubImuData()
{
#ifdef  USE_SIMULATION
    // string sImu_data_file = sConfig_path + "imu_data.txt";
    string sImu_data_file = sConfig_path + "imu_noise_data.txt";
#else
    string sImu_data_file = sConfig_path + "MH_05_imu0.txt";
#endif

    cout << "1 PubImuData start sImu_data_filea: " << sImu_data_file << endl;
    ifstream fsImu;
    fsImu.open(sImu_data_file.c_str());
    if (!fsImu.is_open())
    {
        cerr << "Failed to open imu file! " << sImu_data_file << endl;
        return;
    }

    std::string sImu_line;
    double dStampNSec = 0.0;
    Vector3d vAcc;
    Vector3d vGyr;
    while (std::getline(fsImu, sImu_line) && !sImu_line.empty()) // read imu data
    {
        std::istringstream ssImuData(sImu_line);
        ssImuData >> dStampNSec >> vGyr.x() >> vGyr.y() >> vGyr.z() >> vAcc.x() >> vAcc.y() >> vAcc.z();
        //cout << "Imu t: " << fixed << dStampNSec << " gyr: " << vGyr.transpose() << " acc: " << vAcc.transpose() << endl;
#ifdef  USE_SIMULATION
        pSystem->PubImuData(dStampNSec, vGyr, vAcc);
#else
        pSystem->PubImuData(dStampNSec / 1e9, vGyr, vAcc);
#endif
        usleep(5000*nDelayTimes); //10ms 100hz
    }
    fsImu.close();
}
```

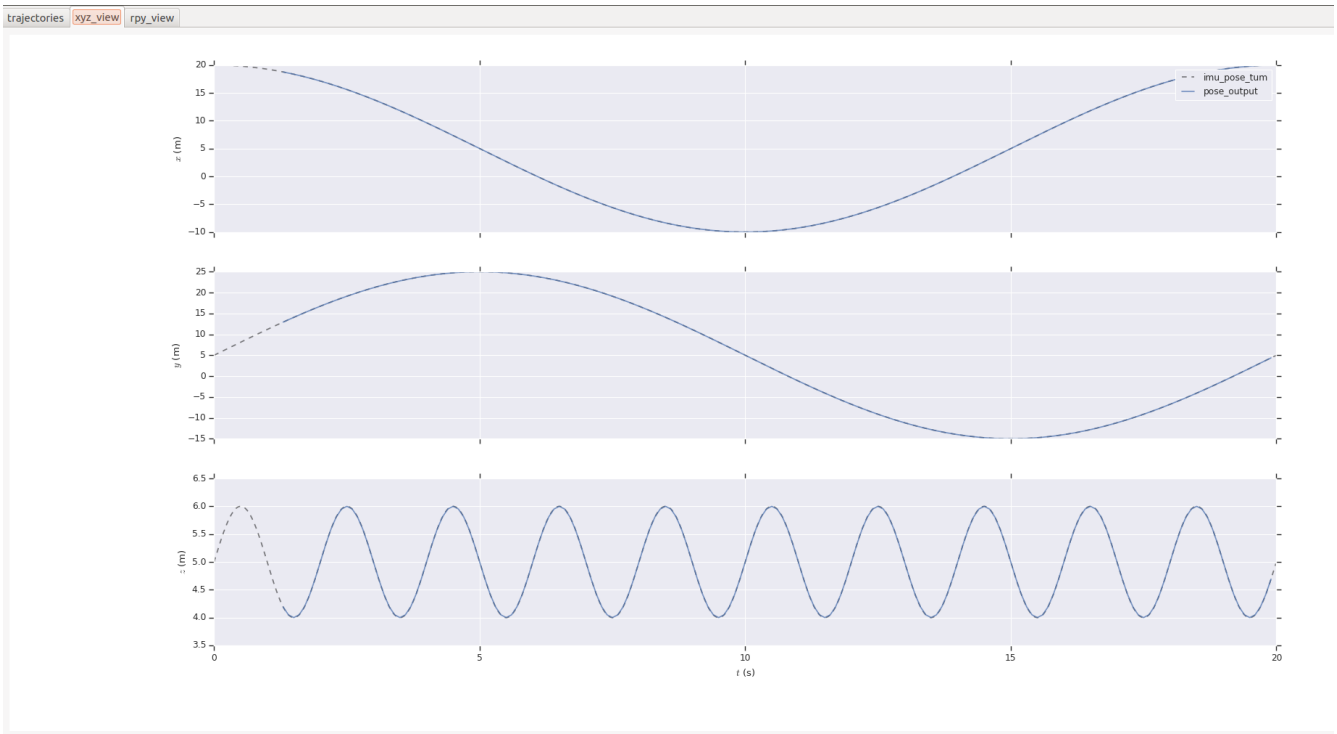
1．使用不含噪声的IMU数据进行测试，并使用evo评估，命令如下：

```
max@max-Inspiron-7559:~/Documents/ch8/VINS-Course/bin$ evo_traj tum imu_pose_tum.txt pose_output.txt --ref=imu_pose_tum.txt -p -a --plot_mode=xz
--------------------------------------------------------------------------------
name:   pose_output
infos:  187 poses, 109.628m path length, 18.600s duration
--------------------------------------------------------------------------------
name:   imu_pose_tum
infos:  2000 poses, 118.967m path length, 19.990s duration
```

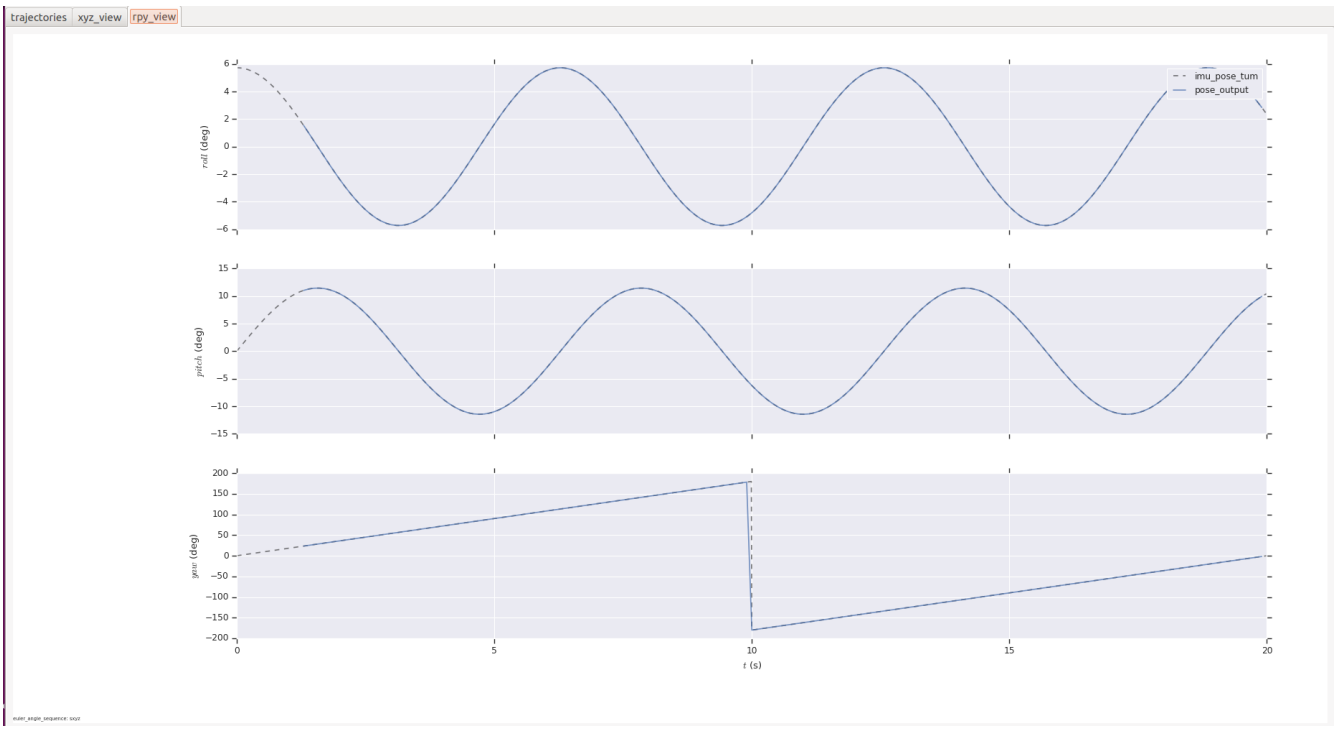
输出轨迹与IMU真实轨迹对比：



x,y,z各方向对比：

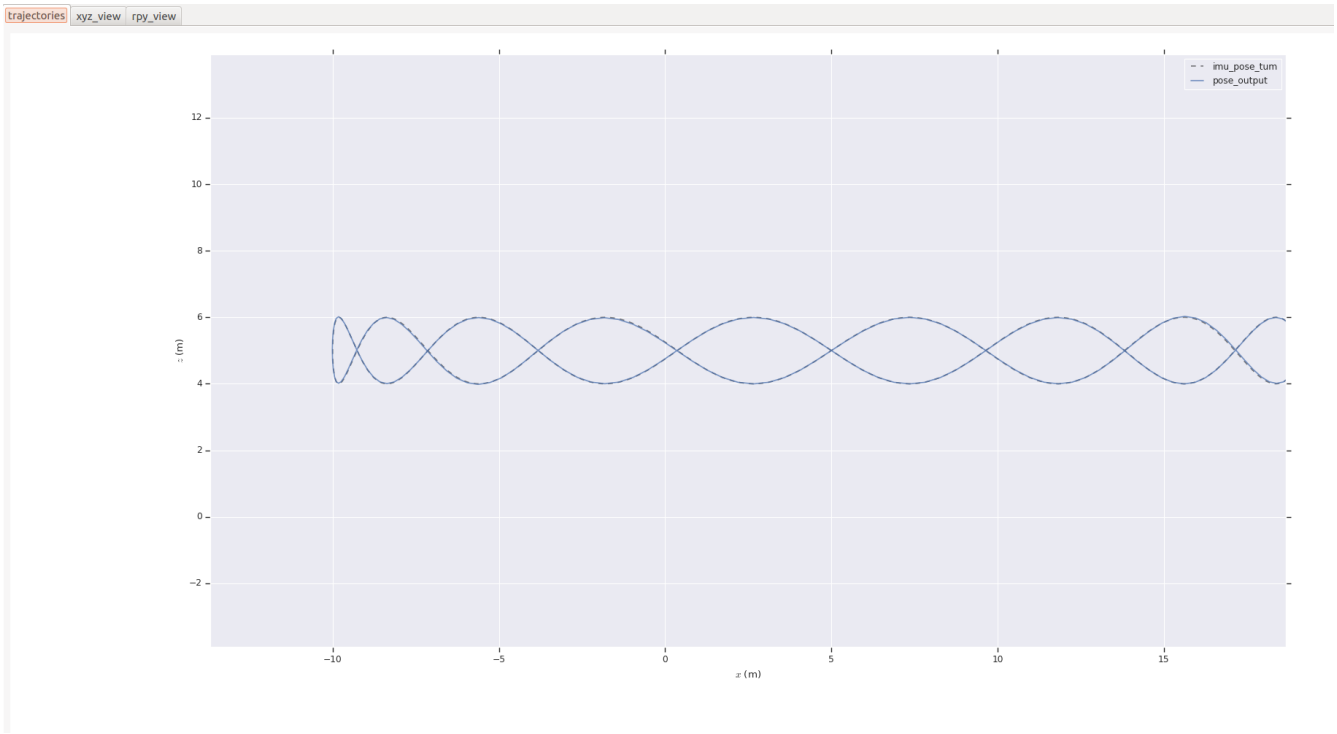


rpy_view:

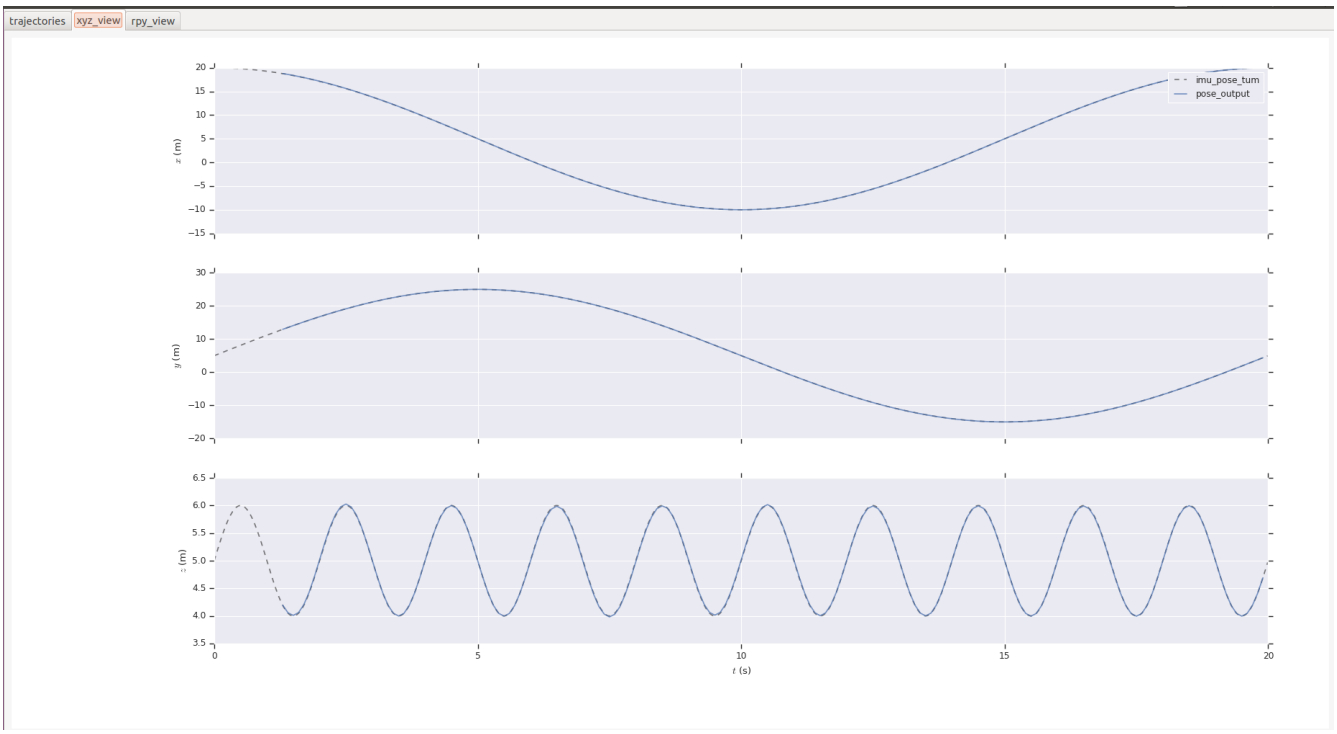


2.使用含噪声的数据测试结果：

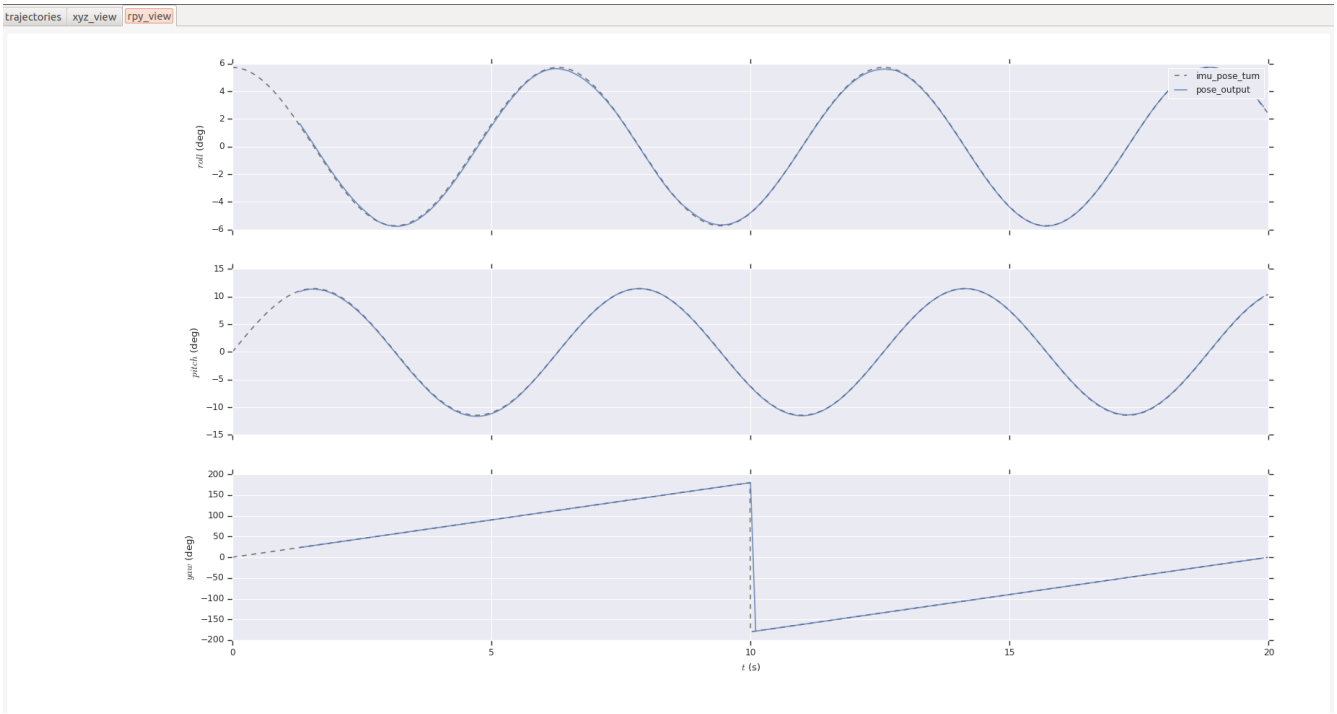
输出轨迹与IMU真实轨迹对比：



x,y,z各方向对比：



rpy_view:



3．更换一组IMU噪声更大的仿真数据进行测试：

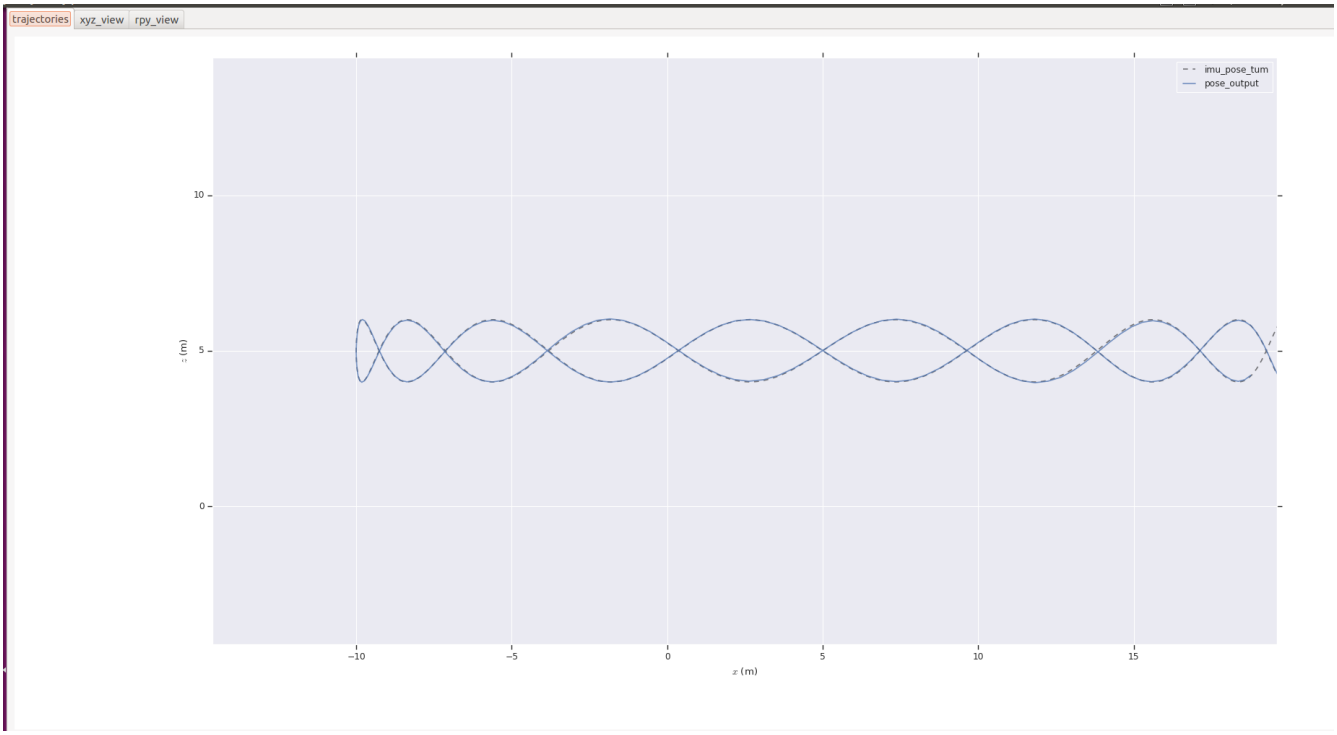
```
// noise
double gyro_bias_sigma = 0.003;//1.0e-5;
double acc_bias_sigma = 0.003;//0.0001;

double gyro_noise_sigma = 0.03;//0.015;    // rad/s
double acc_noise_sigma = 0.03;//0.019;      //  m/(s^2)

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

VINS配置参数：

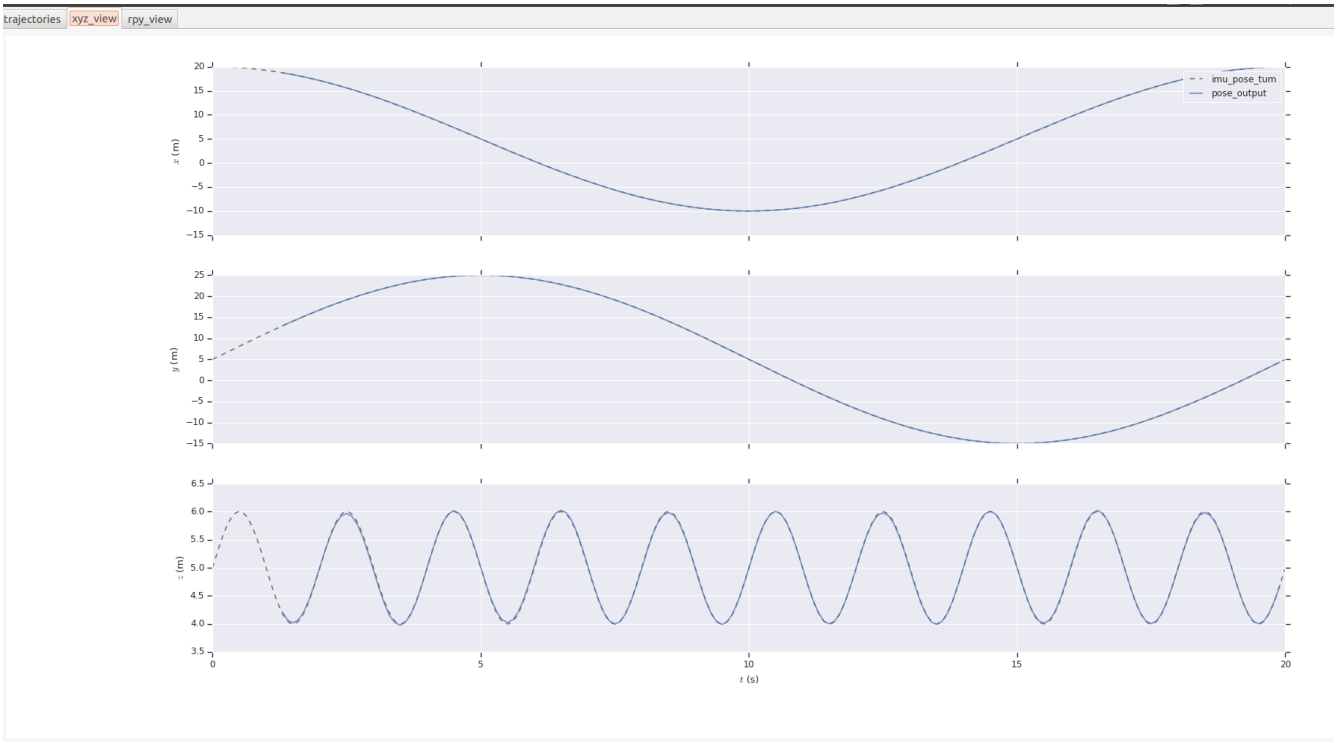
```
#仿真数据集里面的参数就是标准差，vins代码里面用的是方差，所以这里只需要转成离散时间
#imu parameters       The more accurate parameters you provide, the better performance
acc_n: 0.3   #0.03          # accelerometer measurement noise standard deviation. #0.2   0.04
gyr_n: 0.3   #0.03          # gyroscope measurement noise standard deviation.     #0.05  0.004
acc_w: 0.0003  #0.003         # accelerometer bias random work noise standard deviation.  #0.02
gyr_w: 0.0003  #0.003       # gyroscope bias random work noise standard deviation.     #4.0e-5
g_norm: 9.81007     # gravity magnitude
```

测试结果：

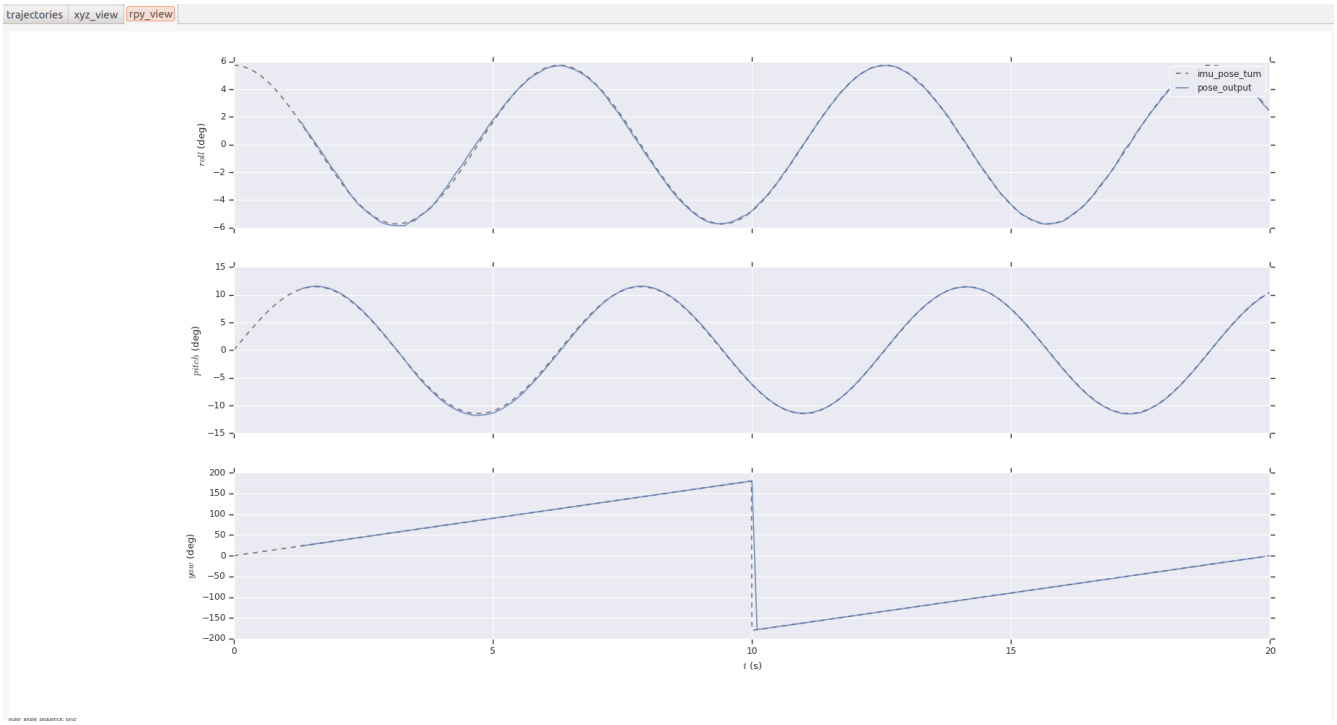
输出轨迹与IMU真实轨迹对比：



x,y,z各方向对比：



rpy_view，方框代表不重合的地方：



TODO:外参估计不准的问题(主要是旋转部分，小的平移不影响)，噪声参数不准的影响（偏大，偏小），陀螺仪噪声估计参数给的偏小，容易影响定位精度，因为比较更相信陀螺仪的数据。