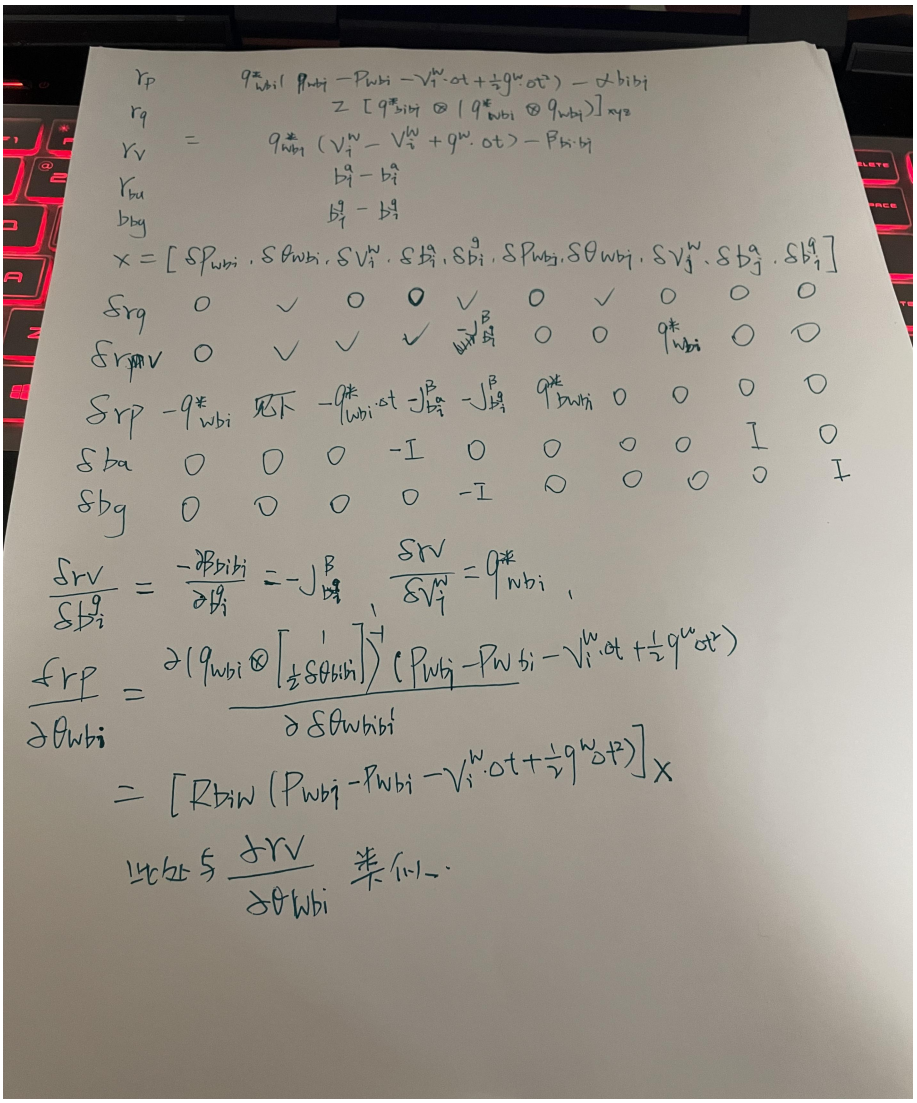
1．推导雅克比矩阵

对号是已推的，剩下的很简单 只有一个 p 对 theta 的求导，参考 q 对 theta 的也很简单。



2．实现代码：

主要是预积分的过程和优化 error 的设计

```
                Eigen::Matrix3d dp_dba = J_.block<3, 3>(INDEX_P, INDEX_A);
                Eigen::Matrix3d dp_dbg = J_.block<3, 3>(INDEX_P, INDEX_G);

                Eigen::Matrix3d dq_dbg = J_.block<3, 3>(INDEX_R, INDEX_G);

                Eigen::Matrix3d dv_dba = J_.block<3, 3>(INDEX_V, INDEX_A);
                Eigen::Matrix3d dv_dbg = J_.block<3, 3>(INDEX_V, INDEX_G);

                //
                // TODO: update pre-integration measurement caused by bias change:

                if(v0->isUpdated())
                {
                    Eigen::Vector3d ba,bg;
                    v0->getDeltaBiases(ba,bg);
                    updateMeasurement(ba,bg);
                }
                //
//                J_.resize(15,30);
//                J_.setZero();
//                J_.block<3, 3>(INDEX_P, INDEX_P) = -ori_i.inverse().matrix();
//                J_.block<3, 3>(INDEX_P, INDEX_R) = Sophus::SO3d::hat(ori_i.inverse().matrix()*(pos_j-pos_i-vel_i*T_+0.5*g_*T_*T_));
//                J_.block<3, 3>(INDEX_P, INDEX_V) = -ori_i.inverse().matrix()*T_;
//                J_.block<3, 3>(INDEX_P, INDEX_A) = -dp_dba;
//                J_.block<3, 3>(INDEX_P, INDEX_G) = -dp_dbg;
//                J_.block<3, 3>(INDEX_P, INDEX_P+15) =  -ori_j.inverse().matrix();
```

```
  //
  // TODO: compute error:
  //
  _error.block<3, 1>(INDEX_P, 0) = Eigen::Vector3d::Zero();
  _error.block<3, 1>(INDEX_R, 0) = Eigen::Vector3d::Zero();
  _error.block<3, 1>(INDEX_V, 0) = Eigen::Vector3d::Zero();
  _error.block<3, 1>(INDEX_A, 0) = Eigen::Vector3d::Zero();
  _error.block<3, 1>(INDEX_G, 0) = Eigen::Vector3d::Zero();

  _error.block<3, 1>(INDEX_P, 0) = ori_i.inverse().matrix()*(pos_j-pos_i-vel_i*T_+0.5*g_*T_*T_)-_measurement.block<3, 1>(INDEX_P, 0);
  _error.block<3, 1>(INDEX_R, 0) = 2*Eigen::Quaterniond(Sophus::SO3d::exp(_measurement.block<3, 1>(INDEX_R, 0)).inverse().matrix()*ori_i.inverse().matrix()*ori_j.
  _error.block<3, 1>(INDEX_V, 0) = ori_i.inverse()*(vel_j-vel_i+g_*T_)-_measurement.block<3, 1>(INDEX_V, 0);
  _error.block<3, 1>(INDEX_A, 0) = b_a_j-b_a_i;
  _error.block<3, 1>(INDEX_G, 0) = b_g_j-b_g_i;
```

## 顶点的更新

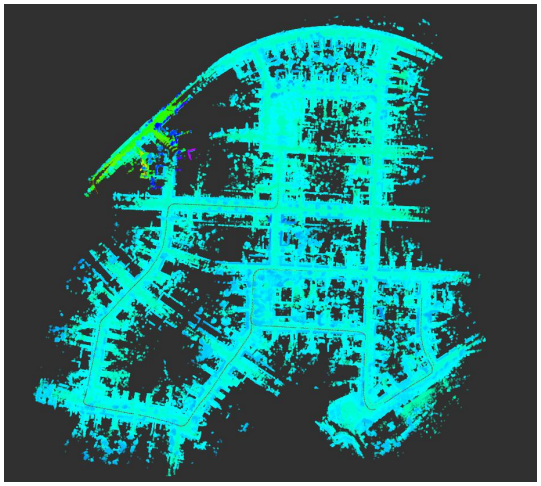
```
    virtual void oplusImpl(const double *update) override {
        //
        // TODO: do update
        //
            _estimate.pos+=Eigen::Vector3d(update[PRVAG::INDEX_POS+0],update[PRVAG::INDEX_POS+1],update[PRVAG::INDEX_POS+2]);
            _estimate.ori=_estimate.ori*Sophus::SO3d::exp(Eigen::Vector3d(update[PRVAG::INDEX_ORI+0],update[PRVAG::INDEX_ORI+1],update[PRVAG:
            _estimate.vel+=Eigen::Vector3d(update[PRVAG::INDEX_VEL+0],update[PRVAG::INDEX_VEL+1],update[PRVAG::INDEX_VEL+2]);
            _estimate.b_a+=Eigen::Vector3d(update[PRVAG::INDEX_B_A+0],update[PRVAG::INDEX_B_A+1],update[PRVAG::INDEX_B_A+2]);
            _estimate.b_g+=Eigen::Vector3d(update[PRVAG::INDEX_B_G+0],update[PRVAG::INDEX_B_G+1],update[PRVAG::INDEX_B_G+2]);
            updateDeltaBiases(Eigen::Vector3d(update[PRVAG::INDEX_B_A+0],update[PRVAG::INDEX_B_A+1],update[PRVAG::INDEX_B_A+2]),
                    Eigen::Vector3d(update[PRVAG::INDEX_B_G+0],update[PRVAG::INDEX_B_G+1],update[PRVAG::INDEX_B_G+2]));
    }
```

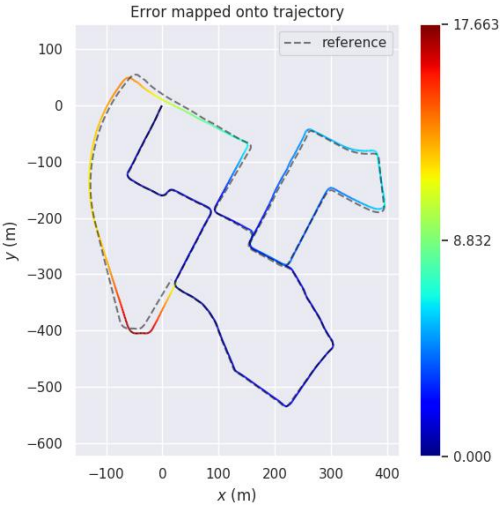
```
void IMUPreIntegrator::UpdateState(void) {
    static double T = 0.0;

    static Eigen::Vector3d w_mid = Eigen::Vector3d::Zero();
    static Eigen::Vector3d a_mid = Eigen::Vector3d::Zero();

    static Sophus::SO3d prev_theta_ij = Sophus::SO3d();
    static Sophus::SO3d curr_theta_ij = Sophus::SO3d();
    static Sophus::SO3d d_theta_ij = Sophus::SO3d();

    static Eigen::Matrix3d dR_inv = Eigen::Matrix3d::Zero();
    static Eigen::Matrix3d prev_R = Eigen::Matrix3d::Zero();
    static Eigen::Matrix3d curr_R = Eigen::Matrix3d::Zero();
    static Eigen::Matrix3d prev_R_a_hat = Eigen::Matrix3d::Zero();
    static Eigen::Matrix3d curr_R_a_hat = Eigen::Matrix3d::Zero();

    //
    // parse measurements:
    //
    // get measurement handlers:
    const IMUData &prev_imu_data = imu_data_buff_.at(0);
    const IMUData &curr_imu_data = imu_data_buff_.at(1);

    // get time delta:
    T = curr_imu_data.time - prev_imu_data.time;
```

```
  // get measurements:
  const Eigen::Vector3d prev_w(
      prev_imu_data.angular_velocity.x - state.b_g_i_.x(),
      prev_imu_data.angular_velocity.y - state.b_g_i_.y(),
      prev_imu_data.angular_velocity.z - state.b_g_i_.z()
  );
  const Eigen::Vector3d curr_w(
      curr_imu_data.angular_velocity.x - state.b_g_i_.x(),
      curr_imu_data.angular_velocity.y - state.b_g_i_.y(),
      curr_imu_data.angular_velocity.z - state.b_g_i_.z()
  );

  const Eigen::Vector3d prev_a(
      prev_imu_data.linear_acceleration.x - state.b_a_i_.x(),
      prev_imu_data.linear_acceleration.y - state.b_a_i_.y(),
      prev_imu_data.linear_acceleration.z - state.b_a_i_.z()
  );
  const Eigen::Vector3d curr_a(
      curr_imu_data.linear_acceleration.x - state.b_a_i_.x(),
      curr_imu_data.linear_acceleration.y - state.b_a_i_.y(),
      curr_imu_data.linear_acceleration.z - state.b_a_i_.z()
  );

// 1. get w_mid:
w_mid = 0.5 * ( prev_w + curr_w );
// 2. update relative orientation, so3:
prev_theta_ij = state.theta_ij_;
d_theta_ij = Sophus::SO3d::exp(w_mid * T);
state.theta_ij_ = state.theta_ij_ * d_theta_ij;
curr_theta_ij = state.theta_ij_;
// 3. get a_mid:
a_mid = 0.5 * ( prev_theta_ij*prev_a + prev_theta_ij*curr_a );
// 4. update relative translation:
state.alpha_ij_+=state.beta_ij_*T+0.5*a_mid*T*T;
// 5. update relative velocity:
state.beta_ij_+=a_mid*T;

//
// TODO: b. update covariance:
//
// 1. intermediate results:
dR_inv = d_theta_ij.inverse().matrix();
prev_R = prev_theta_ij.matrix();
curr_R = curr_theta_ij.matrix();
prev_R_a_hat=prev_R * Sophus::SO3d::hat(prev_a);
curr_R_a_hat=curr_R * Sophus::SO3d::hat(curr_a);


//
// TODO: 2. set up F:
//
// F12 & F32:
F_=MatrixF::Identity();
F_.block<3,3>(INDEX_ALPHA,INDEX_THETA)+=-0.25*T*T*(prev_R_a_hat+curr_R_a_hat*(Eigen::Matrix3d::Identity()-Sophus::SO3d::hat(w_mid)*T));
F_.block<3,3>(INDEX_BETA,INDEX_THETA)+=-0.5*T*(prev_R_a_hat+curr_R_a_hat*(Eigen::Matrix3d::Identity()-Sophus::SO3d::hat(w_mid)*T));
// F14 & F34:
F_.block<3,3>(INDEX_ALPHA,INDEX_BETA)+=Eigen::Matrix3d::Identity()*T;
F_.block<3,3>(INDEX_ALPHA,INDEX_B_A)+=-0.25*T*T*(prev_R+curr_R);
// F15 & F35:
F_.block<3,3>(INDEX_ALPHA,INDEX_B_G)+=0.25*T*T*T*curr_R*Sophus::SO3d::hat(curr_a);
F_.block<3,3>(INDEX_THETA,INDEX_B_G)+=-Eigen::Matrix3d::Identity()*T;
F_.block<3,3>(INDEX_BETA,INDEX_B_A)+=-0.5*T*(prev_R+curr_R);
F_.block<3,3>(INDEX_BETA,INDEX_B_G)+=0.5*T*T*curr_R*Sophus::SO3d::hat(curr_a);

// F22:
F_.block<3,3>(INDEX_THETA,INDEX_THETA)+=Eigen::Matrix3d::Identity()-Sophus::SO3d::hat(w_mid)*T;
```

```
B_=MatrixB::Zero();
B_.block<3,3>(INDEX_ALPHA,INDEX_M_ACC_PREV)=0.25*prev_R*T*T;
B_.block<3,3>(INDEX_ALPHA,INDEX_M_GYR_PREV)=-0.125*curr_R*T*T*T*Sophus::SO3d::hat(curr_a);
B_.block<3,3>(INDEX_ALPHA,INDEX_M_ACC_CURR)=0.25*curr_R*T*T;
B_.block<3,3>(INDEX_ALPHA,INDEX_M_GYR_CURR)=-0.125*curr_R*T*T*T*Sophus::SO3d::hat(curr_a);

B_.block<3,3>(INDEX_THETA,INDEX_M_GYR_PREV)=0.5*Eigen::Matrix3d::Identity()*T;
B_.block<3,3>(INDEX_THETA,INDEX_M_GYR_CURR)=0.5*Eigen::Matrix3d::Identity()*T;

B_.block<3,3>(INDEX_BETA,INDEX_M_ACC_PREV)=0.5*prev_R*T;
B_.block<3,3>(INDEX_BETA,INDEX_M_GYR_PREV)=-0.25*curr_R*T*T*Sophus::SO3d::hat(curr_a);
B_.block<3,3>(INDEX_BETA,INDEX_M_ACC_CURR)=0.5*curr_R*T;
B_.block<3,3>(INDEX_BETA,INDEX_M_GYR_CURR)=-0.25*curr_R*T*T*Sophus::SO3d::hat(curr_a);

B_.block<3,3>(INDEX_B_A,INDEX_R_ACC_PREV)=Eigen::Matrix3d::Identity()*T;
B_.block<3,3>(INDEX_B_G,INDEX_R_GYR_PREV)=Eigen::Matrix3d::Identity()*T;

// G11 & G31:
// G12 & G32:
// G13 & G33:
// G14 & G34:

// TODO: 4. update P_:

P_=F_*P_*F_.transpose()+B_*Q_*B_.transpose();
// ;
// TODO: 5. update Jacobian:
//
J_=F_*J_;
```

Gnss、loop、imu_pre、odom_pre 的开关在 config 中






有 imu 积分时

| | |
|---|---|
| max | 17.577519 |
| mean | 4.669541 |
| median | 3.168825 |
| min | 0.000001 |
| rmse | 6.336322 |
| sse | 59259.885910 |
| std | 4.283032 |

无 imu 积分时

| | |
|---|---|
| max | 17.663317 |

| | |
|---|---|
| mean | 5.082181 |
| median | 3.510857 |
| min | 0.000001 |
| rmse | 6.896013 |
| sse | 73234.693898 |
| std | 4.661162 |

有 imu 还是 rmse 稍小点，看了 error 随时间的变化，没发现加 imu 应该有的路段会平滑的现象。之后加了码盘再试试看

3、推导编码器有了 imu 的经验，也会很简单，跟加速度相关项去掉差不多就一致了