一. 绘制信息矩阵及边缘化

题目：某时刻，SLAM 系统中相机和路标点的观测关系如图 1 所示，其中 ξ 表示相机位姿，L 表示在世界坐标系下时，第 k 个路标被第 i 时刻的相机观测到，重投影误差为 $r(\xi_i, L_k)$。另外，相邻相机之间存在运动约束，如 IMU 或者轮速计等约束。

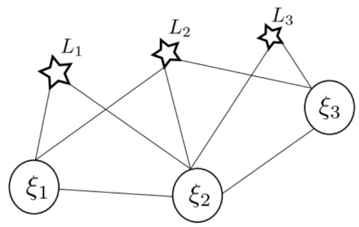

图 1: line chart of lambda

答：这道题就直接在 PPT 上作图回答了，解答过程如图 2 所示，图中 k_i 表示相机位姿，l_i 表示 landmark。

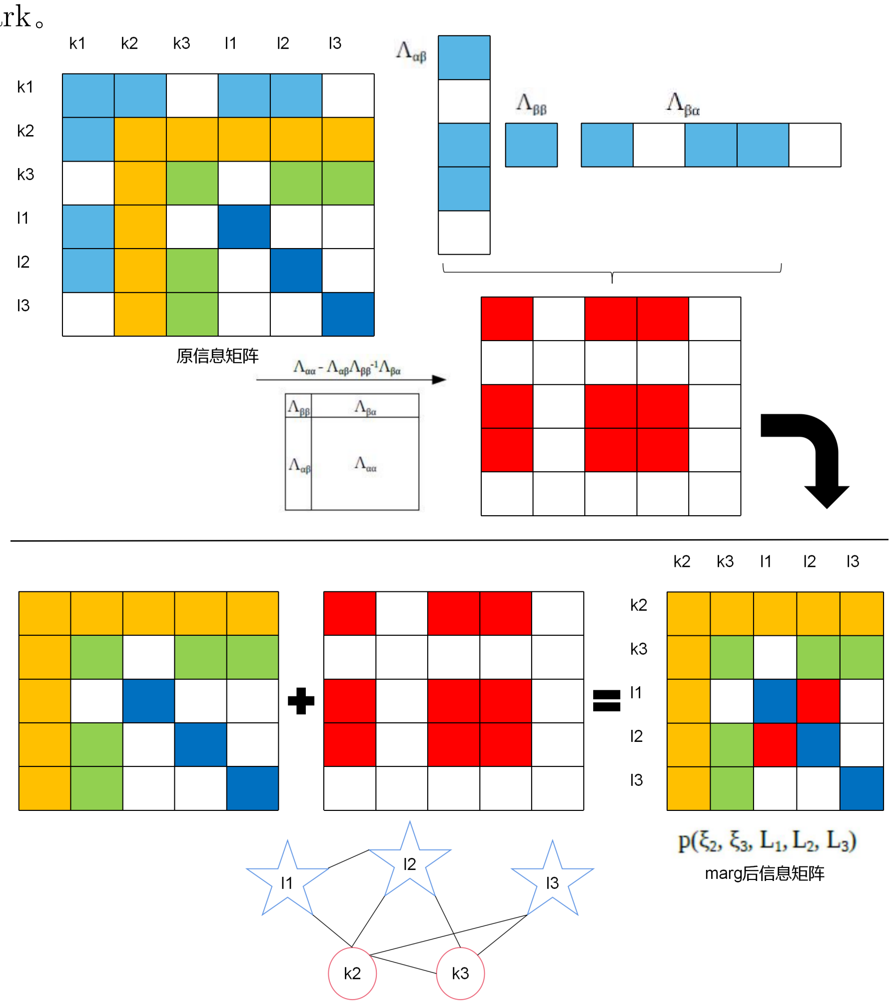

图 2: line chart of lambda

二. 阅读附加材料，证明信息矩阵和协方差矩阵的逆相等

题目：阅读《Relationship between the Hessian and Covariance Matrix for Gaussian Random Variables》。证明信息矩阵和协方差矩阵的逆之间的关系。

答：

信息矩阵 (information martix): 数理统计学中，费希尔信息（英语：Fisher Information；有时称作 information），或称费希尔信息数，通常记作 $\mathcal{I}_X(\theta)$，是衡量观测所得的随机变量 X 携带的关于未知参数 θ 的信息量，其中 X 的概率分布依赖于参数 θ。费希尔信息由统计学家罗纳德 · 费希尔在弗朗西斯 · 伊西德罗 · 埃奇沃思工作的基础上提出，现常用于最大似然估计和贝叶斯统计学中 [1]。

常用于最大似然估计和贝叶斯统计学的话，应该跟我们用的是一个东西了，Fisher information 的定义就是关于变量对数似然函数的二阶导数，所以也就是 Hessian 矩阵，所以 Appendix A 中的 Hessian 就是这个意思吗？虽然不知道这里的 Hessian 是哪里的叫法，可能是原文中？不过题目应该就是证明一个变量概率密度函数的负对数的 Hessian 矩阵等于其协方差矩阵的逆。具体证明过程如下：

假设有高斯随机向量$\boldsymbol{\theta}$，其均值为 θ^*，协方差矩阵为 Σ_θ，则其联合概率密度函数 (joint probability density function) 如式 1 所示：

$$p(\theta) = (2pi)^{-\frac{N_\theta}{2}} |\Sigma_\theta|^{-\frac{1}{2}} exp[-\frac{1}{2}(\theta - \theta^*)^\top \Sigma_\theta^{-1} (\theta - \theta^*)] \tag{1}$$

目标函数可以定义为该联合概率密度函数的负对数 (定义为联合概率密度函数的负对数是因为许多优化问题要求 MAP，然后转换为求负对数最小值，这样梯度下降迭代计算比较方便)，具体形式如式 2 所示：

$$J(\theta) = -lnp(\theta) = \frac{N_\theta}{2} ln 2\pi + \frac{1}{2} ln|\Sigma_\theta| + \frac{1}{2}(\theta - \theta^*)^\top \Sigma_\theta^{-1} (\theta - \theta^*) \tag{2}$$

obviously[2]，该目标函数是变量$\boldsymbol{\theta}$ 的一个二次函数。通过对向量$\boldsymbol{\theta}$ 中不同的两个变量 θ_l 和 $\theta_{l'}$ 求二阶导，可以得到目标函数在 (l, l') 上的 Hessian 矩阵，如式 3 所示：

$$H^{(l,l')}(\theta^*) = \frac{\partial^2 J(\theta)}{\partial \theta_l \partial \theta_{l'}}|_{\theta=\theta^*} = (\Sigma_\theta^{-1})^{(l,l')} \tag{3}$$

所以，目标函数的 Hessian 矩阵 (也就是 Fisher information matrix 或者 information matrix) 等于变量$\boldsymbol{\theta}$ 对应的概率密度函数的协方差矩阵的逆。具体形式如式 4 所示:

$$H(\theta^*) = \Sigma_\theta^{-1} \tag{4}$$

证毕。

三. 补充代码中单目 BA 信息矩阵计算

题目：补充代码中单目 Bundle Adjustment 信息矩阵的计算，验证零空间的维度为 7。

答：代码中首先创建了 10 个相机位姿节点和 20 个 landmarks。其中相机的位姿根据曲线生成，而路标点则在相机位姿周围随机生成，并且每个相机位姿都能看到所以的路边。然后求 Heissian 矩阵，并对其进行 SVD 分解，然后分析它的奇异值，验证是否后七维接近与 0(零空间的维度维 7，有 7 维不可观)

重投影误差对相机位姿和 landmark 坐标的导数这里就不进行推导了，代码里面已经写好了，可以参考《视觉 SLAM14 讲》第 7 讲中的 7.7.3 节。讲解一下代码中构建 Hessian 矩阵的过程，通过两层 for 循环分别遍历所有 landmarks 和 camera poses，在一次循环过程中，对 Hessian 矩阵进行 4 次操作，分别在左上角、右上角、左下角、右下角加入对应的 H_{ii}、H_{ij}、H_{ji} 和 H_{jj}。与下图中的四个红色小块对应，当执行完两层的 for 循环之后，完成 Hessian 矩阵构建。

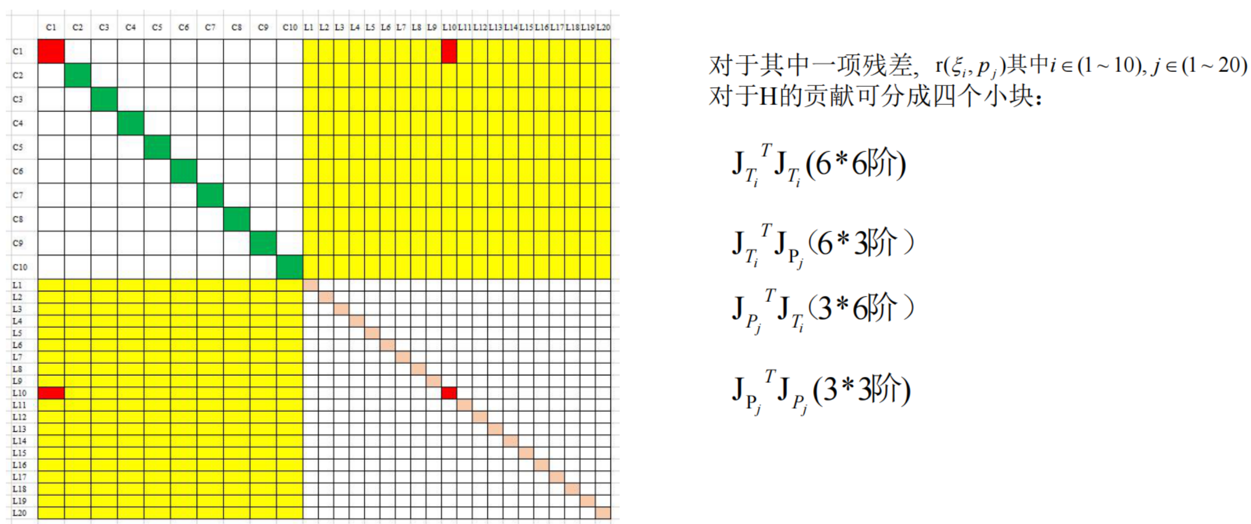

图 3: Hessian 矩阵构建过程

修改的代码如下：

```
for(int j = 0; j < featureNums; ++j)  // 1st for loop:遍历 landmaks
 {
     //... 随机生成landmarks
     for (int i = 0; i < poseNums; ++i) { // 2nd for loop:遍历 cam poses
         Eigen::Matrix3d Rcw = camera_pose[i].Rwc.transpose();
```

```
            Eigen::Vector3d Pc = Rcw * (Pw - camera_pose[i].twc);

            double x = Pc.x();
            double y = Pc.y();
            double z = Pc.z();
            double z_2 = z * z;
            Eigen::Matrix<double,2,3> jacobian_uv_Pc;
            jacobian_uv_Pc<< fx/z, 0 , -x * fx/z_2,
                    0, fy/z, -y * fy/z_2;
            Eigen::Matrix<double,2,3> jacobian_Pj = jacobian_uv_Pc * Rcw;
            Eigen::Matrix<double,2,6> jacobian_Ti;
            jacobian_Ti
                    << -x* y * fx/z_2, (1+ x*x/z_2)*fx, -y/z*fx, fx/z, 0 , -x * fx/z_2,
                       -(1+y*y/z_2)*fy, x*y/z_2 * fy, x/z * fy, 0, fy/z, -y * fy/z_2;
            // 左上 up left
            H.block(i*6,i*6,6,6) += jacobian_Ti.transpose() * jacobian_Ti;
            // 补充完整作业信息矩阵块的计算
            // 右上 up right
                H.block(i * 6,poseNums * 6 + j * 3, 6, 3)
                            += jacobian_Ti.transpose() * jacobian_Pj;
           // 左下 down left
            H.block(poseNums * 6 + j * 3, i * 6, 3, 6)
                        += jacobian_Pj.transpose() * jacobian_Ti;
            // 右下 down right
            H.block(poseNums * 6 + j * 3, poseNums * 6+ j * 3, 3 ,3)
                        += jacobian_Pj.transpose() * jacobian_Pj;
        }
    }
```

程序运行结果最后 10 维的数据如下，可以看到，前三维还比较正常，最后 7 维数据则基本接近于 0，因此可以认为 Hessian 矩阵 NULL Space 的维度为 7。

```
0.00230246
```

```
0.00172459
0.000422374
3.21708e−17
2.06732e−17
1.43188e−17
7.66992e−18
6.08423e−18
6.05715e−18
3.94363e−18
```

四. 一些问题

下面这几个东西 (kalmann 滤波除外) 的 H 矩阵形式都差不多吗 (我现在感觉是这样)，一般又都是梯度下降法迭代求最小值，感觉方法也差不多。他们是否有实质性的差别呢，还是这是技术上的操作不一样 (边缘化、数据结构等)，然后有不同的名字。我有次在知乎上看到 “科技以改名为主”，但是好像又有一些差别，我了解得还不够深入，因此提出这些问题，请梁哥解答一下。

- 最小化重投影误差
- 最大后验概率
- Kalmann Filter 迭代求每一步的最大概率 (基于 Markov 假设)
- Factor Graph 求时间序列以来的最大后验概率

参考文献

[1] https://zh.wikipedia.org/wiki/

[2] http://mark.reid.name/blog/fisher-information-and-log-likelihood.html