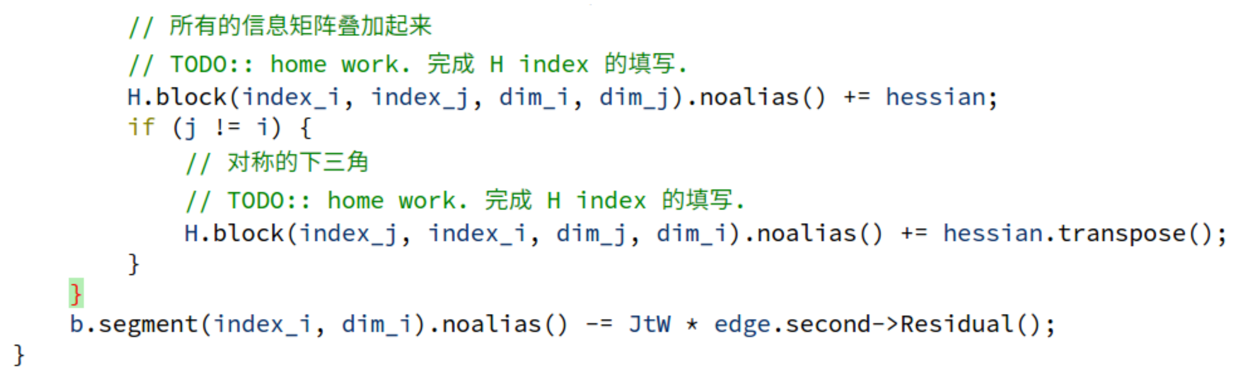
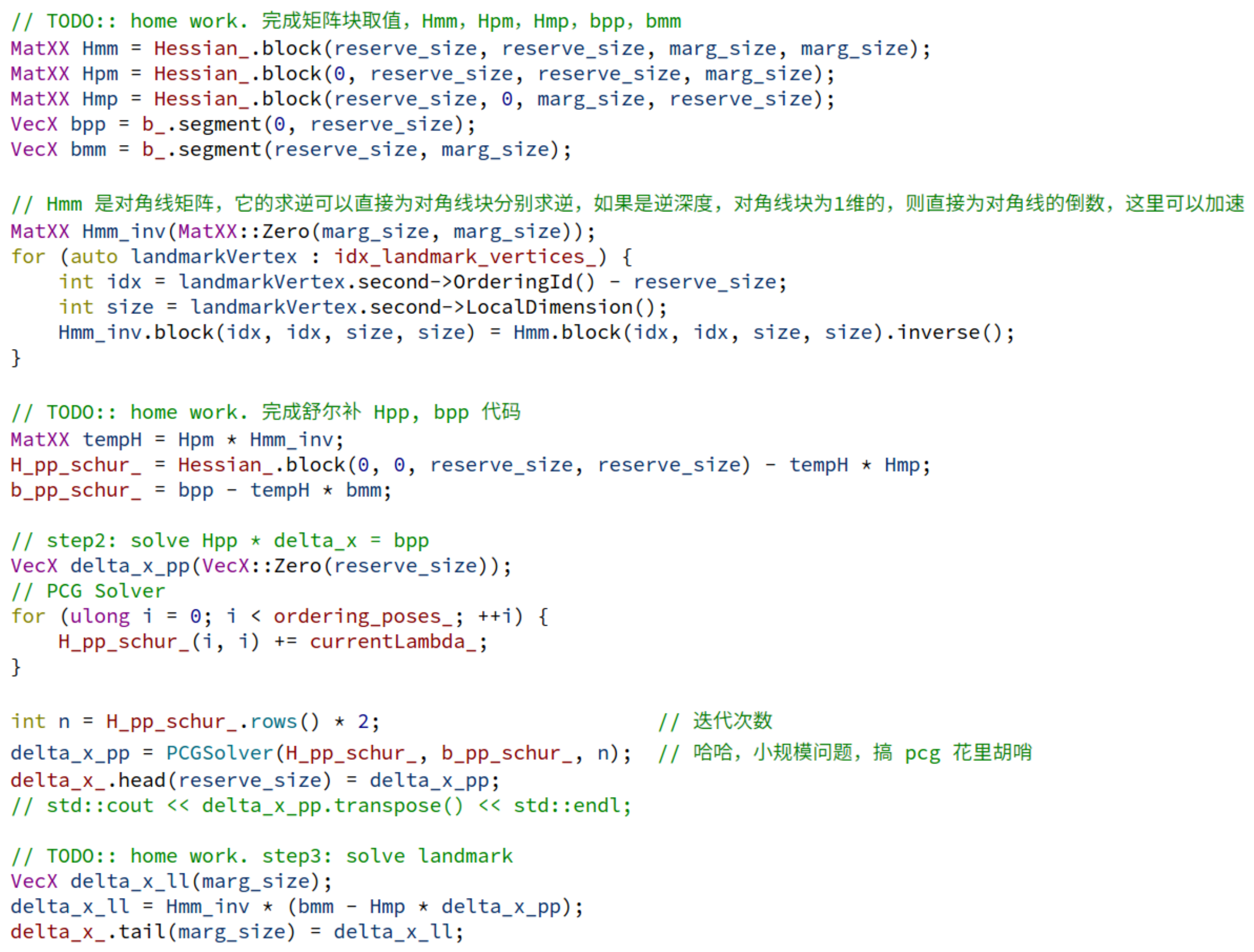
ex.1

1.完成Problem::MakeHessian()中信息矩阵H的计算：



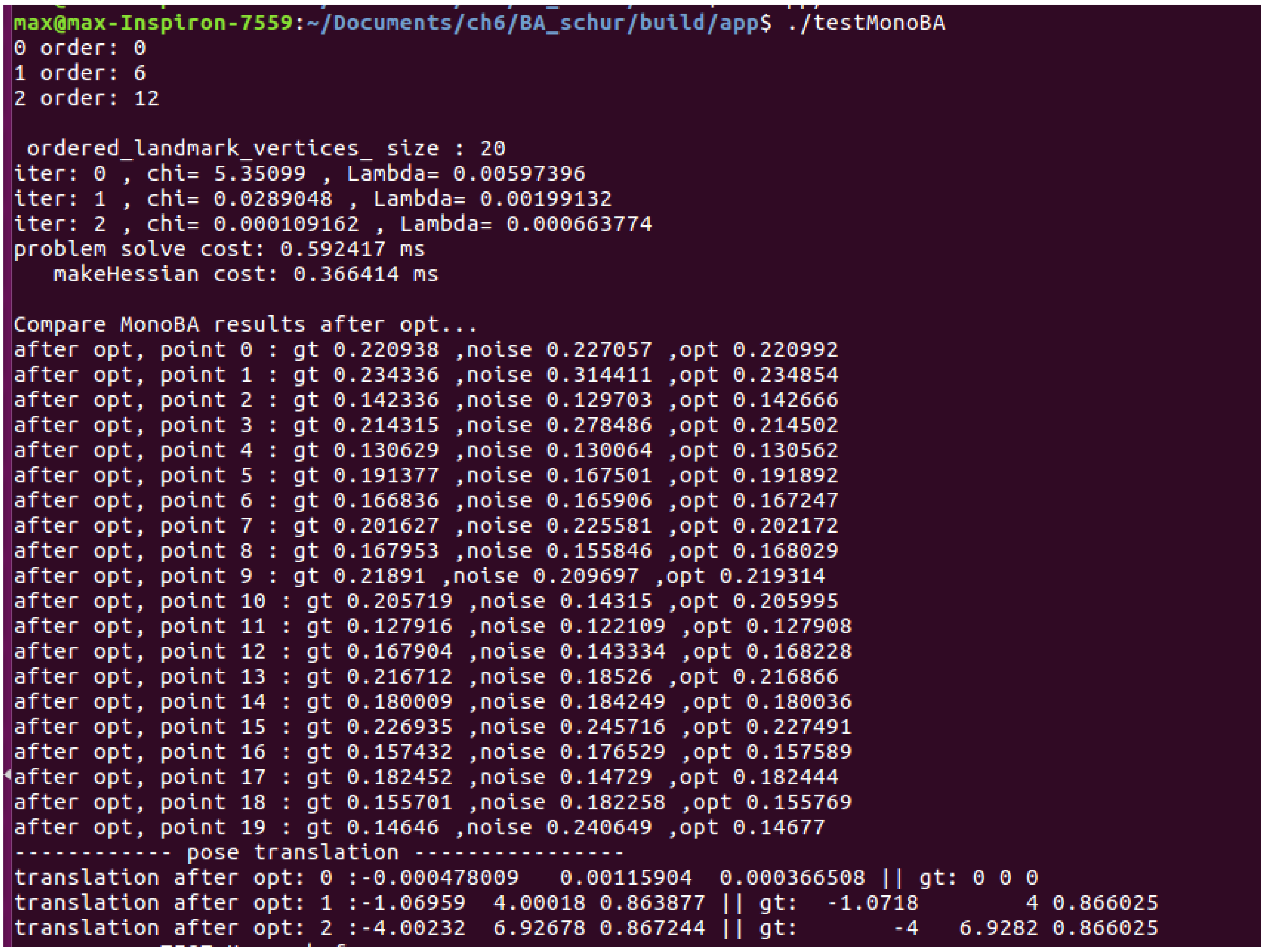
```
            // 所有的信息矩阵叠加起来
            // TODO:: home work. 完成 H index 的填写.
            H.block(index_i, index_j, dim_i, dim_j).noalias() += hessian;
            if (j != i) {
                // 对称的下三角
                // TODO:: home work. 完成 H index 的填写.
                H.block(index_j, index_i, dim_j, dim_i).noalias() += hessian.transpose();
            }
        }
        b.segment(index_i, dim_i).noalias() -= JtW * edge.second->Residual();
    }
```

2．完成Problem::SolveLinearSystem()中slam问题的求解，公式推导参考文档《marginalize推导》：



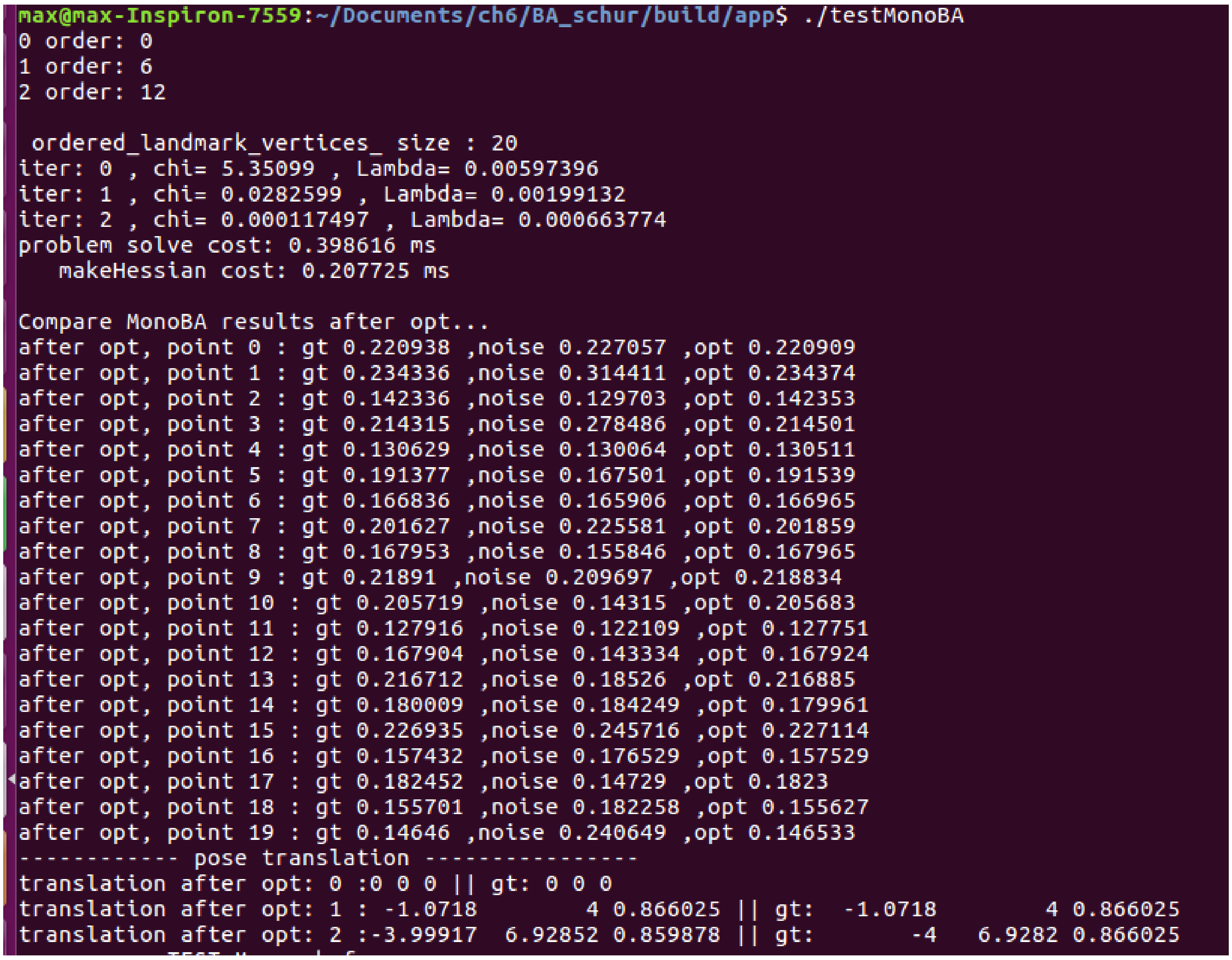
```
// TODO:: home work. 完成矩阵块取值，Hmm，Hpm，Hmp，bpp，bmm
MatXX Hmm = Hessian_.block(reserve_size, reserve_size, marg_size, marg_size);
MatXX Hpm = Hessian_.block(0, reserve_size, reserve_size, marg_size);
MatXX Hmp = Hessian_.block(reserve_size, 0, marg_size, reserve_size);
VecX bpp = b_.segment(0, reserve_size);
VecX bmm = b_.segment(reserve_size, marg_size);

// Hmm 是对角线矩阵，它的求逆可以直接为对角线块分别求逆，如果是逆深度，对角线块为1维的，则直接为对角线的倒数，这里可以加速
MatXX Hmm_inv(MatXX::Zero(marg_size, marg_size));
for (auto landmarkVertex : idx_landmark_vertices_) {
    int idx = landmarkVertex.second->OrderingId() - reserve_size;
    int size = landmarkVertex.second->LocalDimension();
    Hmm_inv.block(idx, idx, size, size) = Hmm.block(idx, idx, size, size).inverse();
}

// TODO:: home work. 完成舒尔补 Hpp, bpp 代码
MatXX tempH = Hpm * Hmm_inv;
H_pp_schur_ = Hessian_.block(0, 0, reserve_size, reserve_size) - tempH * Hmp;
b_pp_schur_ = bpp - tempH * bmm;

// step2: solve Hpp * delta_x = bpp
VecX delta_x_pp(VecX::Zero(reserve_size));
// PCG Solver
for (ulong i = 0; i < ordering_poses_; ++i) {
    H_pp_schur_(i, i) += currentLambda_;
}

int n = H_pp_schur_.rows() * 2;                       // 迭代次数
delta_x_pp = PCGSolver(H_pp_schur_, b_pp_schur_, n);  // 哈哈，小规模问题，搞 pcg 花里胡哨
delta_x_.head(reserve_size) = delta_x_pp;
// std::cout << delta_x_pp.transpose() << std::endl;

// TODO:: home work. step3: solve landmark
VecX delta_x_ll(marg_size);
delta_x_ll = Hmm_inv * (bmm - Hmp * delta_x_pp);
delta_x_.tail(marg_size) = delta_x_ll;
```

输出没有固定前两帧相机位姿的结果：



```
max@max-Inspiron-7559:~/Documents/ch6/BA_schur/build/app$ ./testMonoBA
0 order: 0
1 order: 6
2 order: 12

 ordered_landmark_vertices_ size : 20
iter: 0 , chi= 5.35099 , Lambda= 0.00597396
iter: 1 , chi= 0.0289048 , Lambda= 0.00199132
iter: 2 , chi= 0.000109162 , Lambda= 0.000663774
problem solve cost: 0.592417 ms
   makeHessian cost: 0.366414 ms

Compare MonoBA results after opt...
after opt, point 0 : gt 0.220938 ,noise 0.227057 ,opt 0.220992
after opt, point 1 : gt 0.234336 ,noise 0.314411 ,opt 0.234854
after opt, point 2 : gt 0.142336 ,noise 0.129703 ,opt 0.142666
after opt, point 3 : gt 0.214315 ,noise 0.278486 ,opt 0.214502
after opt, point 4 : gt 0.130629 ,noise 0.130064 ,opt 0.130562
after opt, point 5 : gt 0.191377 ,noise 0.167501 ,opt 0.191892
after opt, point 6 : gt 0.166836 ,noise 0.165906 ,opt 0.167247
after opt, point 7 : gt 0.201627 ,noise 0.225581 ,opt 0.202172
after opt, point 8 : gt 0.167953 ,noise 0.155846 ,opt 0.168029
after opt, point 9 : gt 0.21891 ,noise 0.209697 ,opt 0.219314
after opt, point 10 : gt 0.205719 ,noise 0.14315 ,opt 0.205995
after opt, point 11 : gt 0.127916 ,noise 0.122109 ,opt 0.127908
after opt, point 12 : gt 0.167904 ,noise 0.143334 ,opt 0.168228
after opt, point 13 : gt 0.216712 ,noise 0.18526 ,opt 0.216866
after opt, point 14 : gt 0.180009 ,noise 0.184249 ,opt 0.180036
after opt, point 15 : gt 0.226935 ,noise 0.245716 ,opt 0.227491
after opt, point 16 : gt 0.157432 ,noise 0.176529 ,opt 0.157589
after opt, point 17 : gt 0.182452 ,noise 0.14729 ,opt 0.182444
after opt, point 18 : gt 0.155701 ,noise 0.182258 ,opt 0.155769
after opt, point 19 : gt 0.14646 ,noise 0.240649 ,opt 0.14677
------------ pose translation ----------------
translation after opt: 0 :-0.000478009   0.00115904  0.000366508 || gt: 0 0 0
translation after opt: 1 :-1.06959  4.00018 0.863877 || gt:  -1.0718        4 0.866025
translation after opt: 2 :-4.00232  6.92678 0.867244 || gt:       -4   6.9282 0.866025
```

输出固定前两帧相机位姿的结果：



```
max@max-Inspiron-7559:~/Documents/ch6/BA_schur/build/app$ ./testMonoBA
0 order: 0
1 order: 6
2 order: 12

 ordered_landmark_vertices_ size : 20
iter: 0 , chi= 5.35099 , Lambda= 0.00597396
iter: 1 , chi= 0.0282599 , Lambda= 0.00199132
iter: 2 , chi= 0.000117497 , Lambda= 0.000663774
problem solve cost: 0.398616 ms
   makeHessian cost: 0.207725 ms

Compare MonoBA results after opt...
after opt, point 0 : gt 0.220938 ,noise 0.227057 ,opt 0.220909
after opt, point 1 : gt 0.234336 ,noise 0.314411 ,opt 0.234374
after opt, point 2 : gt 0.142336 ,noise 0.129703 ,opt 0.142353
after opt, point 3 : gt 0.214315 ,noise 0.278486 ,opt 0.214501
after opt, point 4 : gt 0.130629 ,noise 0.130064 ,opt 0.130511
after opt, point 5 : gt 0.191377 ,noise 0.167501 ,opt 0.191539
after opt, point 6 : gt 0.166836 ,noise 0.165906 ,opt 0.166965
after opt, point 7 : gt 0.201627 ,noise 0.225581 ,opt 0.201859
after opt, point 8 : gt 0.167953 ,noise 0.155846 ,opt 0.167965
after opt, point 9 : gt 0.21891 ,noise 0.209697 ,opt 0.218834
after opt, point 10 : gt 0.205719 ,noise 0.14315 ,opt 0.205683
after opt, point 11 : gt 0.127916 ,noise 0.122109 ,opt 0.127751
after opt, point 12 : gt 0.167904 ,noise 0.143334 ,opt 0.167924
after opt, point 13 : gt 0.216712 ,noise 0.18526 ,opt 0.216885
after opt, point 14 : gt 0.180009 ,noise 0.184249 ,opt 0.179961
after opt, point 15 : gt 0.226935 ,noise 0.245716 ,opt 0.227114
after opt, point 16 : gt 0.157432 ,noise 0.176529 ,opt 0.157529
after opt, point 17 : gt 0.182452 ,noise 0.14729 ,opt 0.1823
after opt, point 18 : gt 0.155701 ,noise 0.182258 ,opt 0.155627
after opt, point 19 : gt 0.14646 ,noise 0.240649 ,opt 0.146533
------------ pose translation ----------------
translation after opt: 0 :0 0 0 || gt: 0 0 0
translation after opt: 1 : -1.0718         4 0.866025 || gt:  -1.0718         4 0.866025
translation after opt: 2 :-3.99917  6.92852 0.859878 || gt:        -4   6.9282 0.866025
```

可以看到不固定前两帧相机位姿的时候，ＬＭ求解出来的相机位姿会发生一些微小的漂移。固定前两帧相机位姿后，第一个相机的位姿就固定在原点$(0,0,0)$。