第二次作业

第一题：熟悉Eigen矩阵运算

设线性方程 Ax = b,在 A 为方阵的前提下,请回答以下问题:

1.1 在什么条件下，x有解且唯一

在A为方阵时，需要矩阵A的秩等于方阵的维数时，矩阵方程有唯一解。

$\mathrm{r=m=n}$	$\mathrm{r=n<m}$	$\mathrm{r=m<n}$	$\mathrm{r<m,r<n}$
$\mathrm{R=I}$	$\mathrm{R}=\begin{bmatrix}\mathrm{I}\\0\end{bmatrix}$	$\mathrm{R}=\begin{bmatrix}\mathrm{I} & \mathrm{F}\end{bmatrix}$	$\mathrm{R}=\begin{bmatrix}\mathrm{I} & \mathrm{F}\\0 & 0\end{bmatrix}$
1个解	0或者1个解	∞个解	0或者∞个解

1.2 高斯消元法的原理是什么?

通过将方程A中的未知数，通过方程B用其他的未知数表示出来，之后再带入方程A 就可以达成消去未知数的目的。

1.3 QR 分解的原理是什么?

矩阵QR分解是由Gram–Schmidt正交化推理出来的一种方阵分解形式，矩阵QR分解的计算方法也是以Gram–Schmidt正交化为核心。通过Gram–Schmidt正交化求出正交矩阵Q，再通过 $R=Q^TA$ 得到矩阵R。

1.4 Cholesky 分解的原理是什么?

在Gauss消去与LU的基础上，对特殊的 对称正定矩阵的分解 的加速

$$A=B^TB$$

对称正定阵可以被分解为一个下三角阵及其转置的乘积

$$A = \tilde{L}\tilde{L}^{\mathrm{T}}$$

利用对称性把复杂度减半

STEP 2：求 L_{22} 。

$$L_{22}L_{22}^{\mathrm{T}} = A_{22} - L_{21}L_{21}^{\mathrm{T}}$$

计算 $L_{21}L_{21}^{\mathrm{T}}$ 需要矩阵乘法操作。对于常规矩阵，乘法次数为 $(n-1)^2$ 。而由于对称性，仅需要计算**一半**即可，乘法次数为 $\frac{(n-1)^2}{2}$ 。这也是Cholesky分解的优势所在。

1.5 编程实现 A 为 100 × 100 随机矩阵时,用 QR 和 Cholesky 分解求 x 的程序。

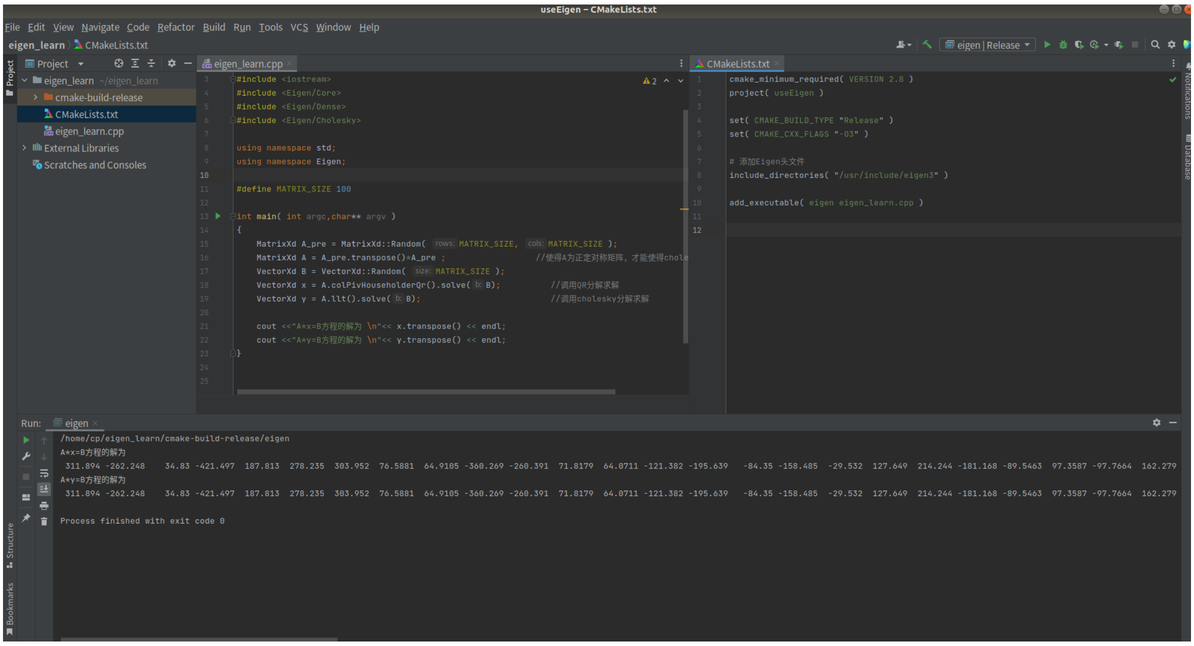
learn_eigen.cpp

```cpp
/*求解 A * x = B 这个方程*/

#include <iostream>
#include <Eigen/Core>
#include <Eigen/Dense>
#include <Eigen/Cholesky>

using namespace std;
using namespace Eigen;

#define MATRIX_SIZE 100

int main( int argc,char** argv )
{
    MatrixXd A_pre = MatrixXd::Random( MATRIX_SIZE, MATRIX_SIZE );
    MatrixXd A = A_pre.transpose()*A_pre ;            //使得A为正定对称矩阵，才能使得cholesky分解成功
    VectorXd B = VectorXd::Random( MATRIX_SIZE );
    VectorXd x = A.colPivHouseholderQr().solve(B);        //调用QR分解求解
    VectorXd y = A.llt().solve(B);                    //调用cholesky分解求解

    cout <<"A*x=B方程的解为 \n"<< x.transpose() << endl;
    cout <<"A*y=B方程的解为 \n"<< y.transpose() << endl;
}


```

CMakeList.txt

```
cmake_minimum_required( VERSION 2.8 )
project( useEigen )

set( CMAKE_BUILD_TYPE "Release" )
set( CMAKE_CXX_FLAGS "-O3" )

# 添加Eigen头文件
include_directories( "/usr/include/eigen3" )

add_executable( eigen eigen_learn.cpp )


```

运行截图



第二题：矩阵论基础

2.1 什么是正定矩阵和半正定矩阵

正定矩阵：给定一个大小为$n \times n$的实对称矩阵A，若对任意长度为n的非零向量x，有$x^TAx > 0$恒成立，则矩阵A是一个正定矩阵。

半正定矩阵：给定一个大小为$n \times n$的实对称矩阵A，若对任意长度为n的非零向量x，有$x^TAx \geqslant 0$恒成立，则矩阵A是一个半正定矩阵。

正定的直观理解：

若给定任意一个半正定矩阵 $A \in \mathbb{R}^{n\times n}$ 和一个向量 $\boldsymbol{x} \in \mathbb{R}^n$ ，则两者相乘得到的向量 $\boldsymbol{y} = A\boldsymbol{x} \in \mathbb{R}^n$ 与向量 $\boldsymbol{x}$ 的夹角恒小于或等于 $\frac{\pi}{2}$. (等价于： $\boldsymbol{x}^T A\boldsymbol{x} \geq 0$.)

2.2 对于方阵 A,它的特征值是什么?特征向量是什么?特征值一定是实数吗?如何计算一个矩阵的特征值?

特征值与特征向量：

我们先来看它的定义，定义本身很简单，假设我们有一个n阶的矩阵A以及一个实数λ，使得我们可以找到一个非零向量x，满足：

$$Ax = \lambda x$$

如果能够找到的话，我们就称λ是矩阵A的特征值，非零向量x是矩阵A的特征向量。

几何理解

从几何方面可以理解为，x与矩阵A做线性变换后，不改变方向只改变大小。

特征值一定是实数吗?：

实矩阵的特征值不一定都是实数，只有实对称矩阵的特征值才保证是实数。另外复矩阵的特征值也可能有实数，同时也有可能是虚数。

如何计算一个矩阵的特征值?

2、n阶方阵A的特征值，就是使齐次线性方程组$(\lambda I - A)\,x = 0$有非零解的λ值，即满足方程$|\lambda I - A| = 0$的λ都是矩阵A的特征值。

3、$|\lambda I - A| = 0 \Leftrightarrow \begin{vmatrix} \lambda - \alpha_{11} & -\alpha_{12} & \dots & -\alpha_{1n} \\ -\alpha_{21} & \lambda - \alpha_{22} & \dots & -\alpha_{2n} \\ \dots & \dots & \dots & \dots \\ -\alpha_{n1} & -\alpha_{n2} & \dots & \lambda - \alpha_{nn} \end{vmatrix} = 0$

2.3么是矩阵的相似性?相似性反映了什么几何意义?

矩阵的相似性的定义:

设 A, B 为 n 阶矩阵，如果有n 阶可逆矩阵 P 存在，使得

$$P^{-1}AP = B$$

则称矩阵 A 与 B 相似，记为 $A \sim B$。

矩阵相似性的几何意义:

同一个运动过程（线性变换）在不同坐标系（不同基）中的表示矩阵（相似矩阵）虽然不一样，但实质上是指的同一个运动过程（线性变换！）

(我为什么要尝试去看懂原理，泪目 本来时间就不充足... 而且看了也是这会儿懂了)

2.4 矩阵一定能对角化吗?什么样的矩阵能保证对角化?不能对角化的矩阵能够形成什么样的形式(Jor-dan 标准形)?

矩阵一定能对角化吗?

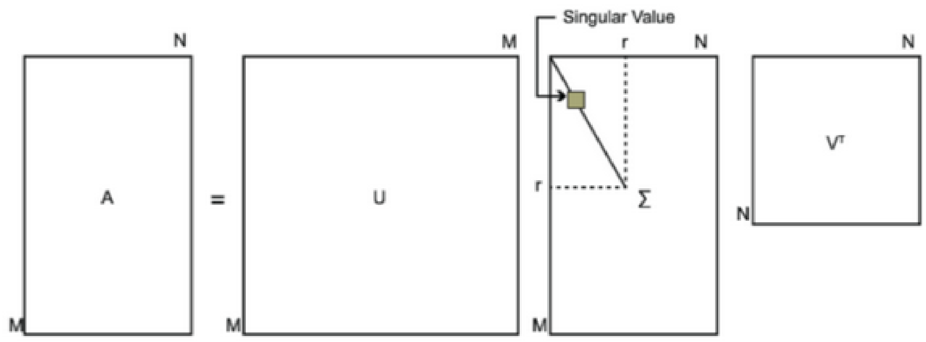
对于一个矩阵来说，不一定存在将其对角化的矩，但是任意一个 $n \times n$矩阵如果存在 n个线性不相关的特征向量，则该矩阵可被对角化。

Uordan标准型的定义

矩阵J 除了主对角线和主对角线上方元素之外，其余都是0，且主对角线上方的对角线的系数若不为0只能为1，且这1的左方和下方的系数（都在主对角线上）有相同的值。易知对角矩阵是一种特殊的Jordan标准型矩阵。

2.5 奇异值分解（SVD）是什么意思?

SVD也是对矩阵进行分解，但是和特征分解不同，SVD并不要求要分解的矩阵为方阵。假设我们的矩阵A是一个m×n的矩阵，那么我们定义矩阵A的SVD为:



2.6 矩阵的伪逆是什么意思(Pseudo inverse)?莫尔——彭多斯逆是如何定义的?怎么计算一个矩阵的伪逆?

伪逆矩阵

伪逆矩阵是逆矩阵的广义形式，由于奇异矩阵或者非方阵的矩阵不存在逆矩阵，但可以有伪逆矩阵（或者说是广义逆矩阵）。

满足$A^LA=E$，但不满足$AA^L=E$的矩阵A^L称为矩阵A的左逆矩阵。同理，满足$AA^R=E$，但不满足$A^RA=E$的矩阵A^R称为矩阵A的左逆矩阵。仅当$m\geq n$时，列满秩，矩阵$A_{m\times n}$有左逆矩阵，$A^L=(A^TA)^{-1}A^T$;仅当$m\leq n$时，行满秩，矩阵$A_{m\times n}$有右逆矩阵，$A^R=A^T(A^TA)^{-1}$；当$m=n, A_{m\times n}$的秩为$r\leq m=n$，对A进行奇异值分解$A=U\Sigma V^T$，A的伪逆矩阵为$A^+=V\Sigma^+U^T$

莫尔——彭多斯逆是如何定义的?

设$A\in C^{m\times n}$，如果 $ G \in C^{n \times m}$满足，

$$\begin{aligned}&(1).\ AGA=A\ ,\\&(2).\ GAG=G\ ,\\&(3).\ (AG)^H=AG\ ,\\&(4).\ (GA)^H=GA\ .\end{aligned}$$

则G为A的$Moore-Penrose$广义逆矩阵。（A^T为A的转置共轭矩阵）

怎么计算一个矩阵的伪逆?

算法核心步骤主要有两步：

1. 计算矩阵$A_{m\times n}$的满秩分解$A=FG$。
2. 求广义逆矩阵，也就是矩阵的伪逆$A^-=G^T(F^TAG^T)^{-1}F^T$。

更具体计算递推公式在《矩阵分析与应用》张贤达这本书中p95页有详细的介绍。

2.7 关于超定方程

暂留

a)

b)

c)

第三题： 几何运算练习

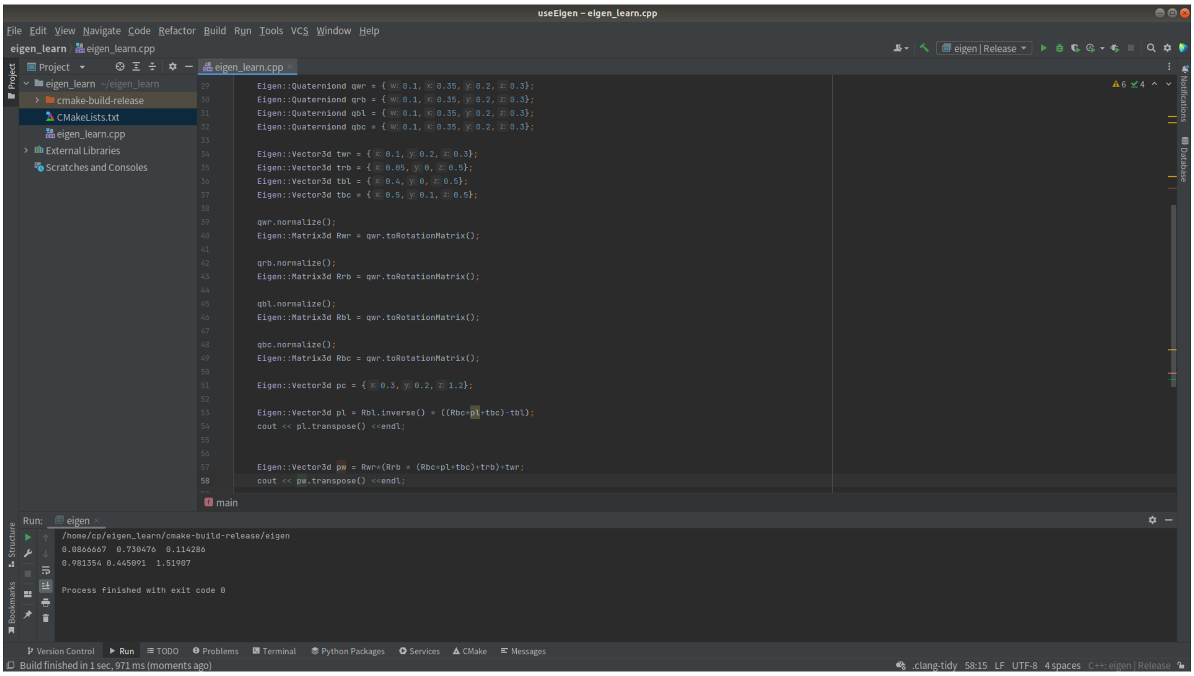
3.1说明一个激光传感器下看到的点应该如何计算它的世界坐标

由坐标转换公式可以退出，在激光传感器下看到的点P的世界坐标为P_L

$$P_L = T_{WR}T_{RB}T_{BL}P_L$$

3.2

没找到代码，按照自己的理解码了一个。。。。



0.0866667 0.730476 0.114286

0.981354 0.445091 1.51907

cpp

```cpp
/*求解 A * x = B 这个方程*/

#include <iostream>
#include <Eigen/Core>
#include <Eigen/Dense>
#include <Eigen/Cholesky>
#include <Eigen/Geometry>


using namespace std;
using namespace Eigen;

#define MATRIX_SIZE 100

int main( int argc,char** argv )
{
   Eigen::Quaterniond qwr = {0.1,0.35,0.2,0.3};
   Eigen::Quaterniond qrb = {0.1,0.35,0.2,0.3};
   Eigen::Quaterniond qbl = {0.1,0.35,0.2,0.3};
   Eigen::Quaterniond qbc = {0.1,0.35,0.2,0.3};

   Eigen::Vector3d twr = {0.1,0.2,0.3};
   Eigen::Vector3d trb = {0.05,0,0.5};
   Eigen::Vector3d tbl = {0.4,0,0.5};
   Eigen::Vector3d tbc = {0.5,0.1,0.5};

   qwr.normalize();
   Eigen::Matrix3d Rwr = qwr.toRotationMatrix();

```

```
    qrb.normalize();
    Eigen::Matrix3d Rrb = qwr.toRotationMatrix();

    qbl.normalize();
    Eigen::Matrix3d Rbl = qwr.toRotationMatrix();

    qbc.normalize();
    Eigen::Matrix3d Rbc = qwr.toRotationMatrix();

    Eigen::Vector3d pc = {0.3,0.2,1.2};

    Eigen::Vector3d pl = Rbl.inverse() * ((Rbc*pl+tbc)-tbl);
    cout << pl.transpose() <<endl;


    Eigen::Vector3d pw = Rwr*(Rrb * (Rbc*pl+tbc)+trb)+twr;
    cout << pw.transpose() <<endl;

}


```

cmake与前文中一样，未做修改

第四题：旋转的表达

4.1 设有旋转矩阵R，证明$R^TR = I$且$detR = \pm 1$

正交

$$[e_1, \quad e_2, \quad e_3]\begin{bmatrix}a_1\\a_2\\a_3\end{bmatrix} = [e'_1, \quad e'_2, \quad e'_3]\begin{bmatrix}a'_1\\a'_2\\a'_3\end{bmatrix}$$

（markdown这个公式编辑，有点劝退啊...是我的打开方式不对吗？）

step1

由于是同一个向量，在对应基表示下相等：

$$[e_1, e_2, e_3]\begin{bmatrix} a \\ b \\ c \end{bmatrix} = [e_1', e_2', e_3']\begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$$

左乘 $[e_1, e_2, e_3]^T$ **，左式由于基向量正交所以得到单位阵，**

$$[e_1, e_2, e_3]^T[e_1, e_2, e_3]\begin{bmatrix} a \\ b \\ c \end{bmatrix} = [e_1, e_2, e_3]^T[e_1', e_2', e_3']\begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} e_1^T e_1' & e_1^T e_2' & e_1^T e_3' \\ e_2^T e_1' & e_2^T e_2' & e_2^T e_3' \\ e_3^T e_1' & e_3^T e_2' & e_3^T e_3' \end{bmatrix}\begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = R\begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$$

step2

左乘 $[e_1', e_2', e_3']^T$ **，右式由于基向量正交所以得到单位阵，**

$$[e_1', e_2', e_3']^T[e_1, e_2, e_3]\begin{bmatrix} a \\ b \\ c \end{bmatrix} = [e_1', e_2', e_3']^T[e_1', e_2', e_3']\begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$$

$$\begin{bmatrix} e_1'^T e_1 & e_1'^T e_2 & e_1'^T e_3 \\ e_2'^T e_1 & e_2'^T e_2 & e_2'^T e_3 \\ e_3'^T e_1 & e_3'^T e_2 & e_3'^T e_3 \end{bmatrix}\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$$

$$R^{-1}\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix}$$

step3

当我们对 R 求转置时，可得 $R^T = R^{-1}$

$$R^T = \begin{bmatrix} e_1^T e_1' & e_1^T e_2' & e_1^T e_3' \\ e_2^T e_1' & e_2^T e_2' & e_2^T e_3' \\ e_3^T e_1' & e_3^T e_2' & e_3^T e_3' \end{bmatrix}^T = \begin{bmatrix} e_1'^T e_1 & e_1'^T e_2 & e_1'^T e_3 \\ e_2'^T e_1 & e_2'^T e_2 & e_2'^T e_3 \\ e_3'^T e_1 & e_3'^T e_2 & e_3'^T e_3 \end{bmatrix} = R^{-1}$$

可简述为两步 1 分别用基向量左乘 得到R和R^{-1},将R转置后 可以得到 R^T=R^{-1}

detR=1

因为矩阵的列向量都是模长为1的向量同时列向量都正交

4.2 设有四元数 q,我们把虚部记为 ε,实部记为 η,那么 q = (ε, η)。请说明 ε 和 η 的维度

分别为三维和一维

4.3

视觉14讲 P59

令$q_1 = [x_1, y_1, z_1, w_1]$, $q_2 = [x_2, y_2, z_2, w_2]$，其中$w_1, w_2$为实部

$$\begin{aligned}
q_1 \cdot q_2 = &(w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2) i \\
&+ (w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2) j \\
&+ (w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2) k \\
&+ w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2
\end{aligned}$$

$$\begin{aligned}
q_1^+ q_2 &= \begin{bmatrix} w_1 & -z_1 & y_1 & x_1 \\ z_1 & w_1 & -x_1 & y_1 \\ -y_1 & x_1 & w_1 & z_1 \\ -x_1 & -y_1 & -z_1 & w_1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ w_2 \end{bmatrix} \\
&= \begin{bmatrix} w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2 \\ w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2 \\ w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2 \\ w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \end{bmatrix} \\
&= (w_1 x_2 + x_1 w_2 + y_1 z_2 - z_1 y_2) i \\
&+ (w_1 y_2 - x_1 z_2 + y_1 w_2 + z_1 x_2) j \\
&+ (w_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 w_2) k \\
&+ w_1 w_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \\
&= q_1 \cdot q_2
\end{aligned}$$

所以：$q_1 \cdot q_2 = q_1^+ q_2$

同理可证：$q_1 \cdot q_2 = q_2^\oplus q_1$

这里用到的证明方式和前面R为可逆矩阵有点类似，都是先计算出来再做对比。

第五题：罗德里格斯公式的证明

step1

对v进行向量分解： $v = v\perp + v//$

由点乘的投影几何意义可得： $v// = (v \cdot k)k$ (v・k为标量，所以再乘k得到一个矢量)

根据向量减法可得： $v\perp = v - v//$

由旋转过程平行向量不变得： vrot //= v//

为计算方便，对vrot⊥进行向量分解：vrot⊥=a+b

由图中的向量关系可得：b= cosΘ v⊥ a=sinΘ k x v

(b可直接由图看出，而a则相对复杂。因为a是vrot⊥在w方向上的分量，所以我们必须用一矢量来表示。已知v⊥和vrot⊥模长相等，所以可用sinΘk x v⊥来表示a。

但是我们先前说过要找出v和vrot的关系，所以要用v去表示。我们发现sinα|v| =|v⊥|，所以sinΘk x v⊥=sinΘ k x v 。其中α为v与k的夹角)

综上可得：vrot = vrot⊥+vrot //

= a + b + v//

= sinΘ k x v + cosΘ v⊥ + (v ・ k)k

= sinΘ k x v + cosΘ (v - v//)+ (v ・ k)k

= sinΘ k x v + cosΘ (v - (v ・ k)k)+ (v ・ k)k

= cosΘ v + (1 - cosΘ)(v ・ k)k + sinΘ k x v

//还没理解通。。。脑细胞已经死完了

step2

$$\begin{bmatrix} (\mathbf{k}\times\mathbf{v})_x \\ (\mathbf{k}\times\mathbf{v})_y \\ (\mathbf{k}\times\mathbf{v})_z \end{bmatrix} = \begin{bmatrix} k_y v_z - k_z v_y \\ k_z v_x - k_x v_z \\ k_x v_y - k_y v_x \end{bmatrix} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}.$$

所以，cosΘ v + (1 - cosΘ)(v ・ k)k + sinΘ k x v

= cosΘ v + (1 - cosΘ)kkτv + sinΘKv

=(cosΘ I + (1 - cosΘ)kkτ + sinΘK) v

=Rv

所以，旋转矩阵R=cosΘ I + (1 - cosΘ)kkτ + sinΘK，其中I为单位矩阵。

证明：

$$\begin{aligned} RR^T &= (cos\theta I + (1-cos\theta)nn^T + sin\theta n^{\times})(cos\theta I + (1-cos\theta)nn^T + sin\theta n^{\times})^T && (13) \\ &= (cos\theta I + (1-cos\theta)nn^T + sin\theta n^{\times})(cos\theta I + (1-cos\theta)nn^T + sin\theta (n^{\times})^T) && (14) \\ &= cos^2\theta I + (1-cos\theta)^2 nn^T nn^T + sin^2\theta n^{\times}(n^{\times})^T + 2cos\theta(1-cos\theta)nn^T && (15) \\ &\quad + sin\theta cos\theta(n^{\times} + n^{\times T}) + (1-cos\theta)sin\theta nn^T((n^{\times})^T + (n^{\times})) \\ &= I + ((1-cos\theta)^2 + 2cos\theta - 2cos^2\theta)nn^T + sin^2\theta n^{\times}(n^{\times})^T && (16) \\ &= I + sin^2\theta(nn^T + n^{\times}(n^{\times})^T) && (17) \end{aligned}$$

其中$n^{\times}$是反对称矩阵，因此$n^{\times T} + n^{\times} = 0$,

其中,设$n = [a, b, c]^T$,则

$$nn^T = \begin{bmatrix} a^2 & ab & ac \\ ab & b^2 & bc \\ ac & b^c & c^2 \end{bmatrix}$$

$$n^{\times} n^{\times} = \begin{bmatrix} a^2-1 & ab & ac \\ ab & b^2-1 & bc \\ ac & b^c & c^2-1 \end{bmatrix}$$

所以：$nn^T + n^{\times}(n^{\times})^T) = nn^T - n^{\times}(n^{\times})) = 0$

所以：$RR^T = I$

因此：$R^{-1} = R^T$

第六题：四元数运算性质的验证

$$p' = qpq^{-1} = q(pq^{-1}) = q(q^{-1^{\oplus}}p) = q^{+}q^{-1^{\oplus}}p$$

$$Q = q^{+}q^{-1^{\oplus}} = \begin{bmatrix} s1 + v^{\times} & v \\ -v^{T} & s \end{bmatrix} \begin{bmatrix} s1 + v^{\times} & -v \\ v^{T} & s \end{bmatrix}$$

$$= \begin{bmatrix} s^2 1 + 2sv^{\times} + v^{\times}v^{\times} & -sv - v^{\times}v + sv \\ -sv^{T} - v^{T}v^{\times} + sv^{T} & s^2 + v^{T}v \end{bmatrix} = \begin{bmatrix} (s1 + v^{\times})^2 & -v^{\times}v \\ -v^{T}v^{\times} & 1 \end{bmatrix}$$

$$Q = \begin{bmatrix} (s1 + v^{\times})^2 & 0 \\ 0 & 1 \end{bmatrix}$$

看懂了，后续有时间补一个手推过程

第七题：熟悉 C++11

题目代码

```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  
5  using namespace std;
6  
7  class A {
8  public:
9      A(const int& i ) : index(i) {};
10     int index = 0;
11 };
12 
13 int main() {
14     A a1(3), a2(5), a3(9);
15     vector<A> avec{a1, a2, a3};
16     std::sort(avec.begin(), avec.end(), [](const A&a1, const A&a2) {return a1.index<a2.index;});
17     for ( auto& a: avec ) cout<<a.index<<" ";
18     cout<<endl;
19     return 0;
20 }
21 
```

第9行使用了初始化列表的方式初始化了字段，这种方式也出现在了ORB-SLAM2源码中

```
A(const int& i ) : index(i) {};

//等价于

A(const int& i )
{
    index = i;
}

```

第15行：使用了初始化列表来初始化对象： C++11 把初始化列表的概念绑定到了类型上，并将其称之为std::initializer_list，允许构造函数或其他函数像参数一样使用初始化列表，这就为类对象的初始化与普通数组和 POD 的初始化方法提供了统一的桥梁。

第16行：使用了lambda表达式来比较元素大小，其中： `const A&a1, const A&a2` 是参数列表， `return a1.index<a2.index;` 是函数体，返回值是布尔型的大小比较结果。

第17行：for(atuo& a: avec) 自动类型推导，使用了auto关键字，可以根据a获得的值自动推断出a的类型。

第17行：C++引入了基于范围的for循环，不用下标就能访问元素；

参考链接

1.1

https://blog.csdn.net/zxnzjccmily/article/details/125996567?spm=1001.2101.3001.6650.3&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-125996567-blog-78685462.pc_relevant_multi_platform_featuressortv2dupreplace&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-3-125996567-blog-78685462.pc_relevant_multi_platform_featuressortv2dupreplace&utm_relevant_index=5

https://zhuanlan.zhihu.com/p/37351063

1.2

https://www.cnblogs.com/xiangqi/p/11138938.html#:~:text=%E9%A6%96%E5%85%88%E4%BA%86%E8%A7%A3%E9%AB%98%E6%96%AF%E6%B6%88%E5%85%83%E6%B3%95%E7%9A%84%E5%8E%9F%E7%90%86%E5%92%8C%E6%A0%B8%E5%BF%83%EF%BC%9A,%E5%8E%9F%E7%90%86%EF%BC%9A%20%E6%B6%88%E5%85%83%E6%B3%95%E6%98%AF%E5%B0%86%E6%96%B9%E7%A8%8B%E7%BB%84%E4%B8%AD%E7%9A%84%E4%B8%80%E6%96%B9%E7%A8%8B%E7%9A%84%E6%9C%AA%E7%9F%A5%E6%95%B0%E7%94%A8%E5%90%AB%E6%9C%89%E5%8F%A6%E4%B8%80%E6%9C%AA%E7%9F%A5%E6%95%B0%E7%9A%84%E4%BB%A3%E6%95%B0%E5%BC%8F%E8%A1%A8%E7%A4%BA%EF%BC%8C%E5%B9%B6%E5%B0%86%E5%85%B6%E4%BB%A3%E4%BA%BA%E5%88%B0%E5%8F%A6%E4%B8%80%E6%96%B9%E7%A8%8B%E4%B8%AD%EF%BC%8C%E8%BF%99%E5%B0%B1%E6%B6%88%E5%8E%BB%E4%BA%86%E4%B8%80%E6%9C%AA%E7%9F%A5%E6%95%B0%EF%BC%8C%E5%BE%97%E5%88%B0%E4%B8%80%E8%A7%A3%EF%BC%9B%E6%88%96%E5%B0%86%E6%96%B9%E7%A8%8B%E7%BB%84%E4%B8%AD%E7%9A%84%E4%B8%80%E6%96%B9%E7%A8%8B%E5%80%8D%E4%B9%98%E6%9F%90%E4%B8%AA%E5%B8%B8%E6%95%B0%E5%8A%A0%E5%88%B0%E5%8F%A6%E5%A4%96%E4%B8%80%E6%96%B9%E7%A8%8B%E4%B8%AD%E5%8E%BB%EF%BC%8C%E4%B9%9F%E5%8F%AF%E8%BE%BE%E5%88%B0%E6%B6%88%E5%8E%BB%E4%B8%80%E6%9C%AA%E7%9F%A5%E6%95%B0%E7%9A%84%E7%9B%AE%E7%9A%84%E3%80%82

1.3

https://blog.csdn.net/u010945683/article/details/45972819

1.4

https://zhuanlan.zhihu.com/p/387603571

1.5

https://blog.csdn.net/weixin_41074793/article/details/84241776?spm=1001.2101.3001.6661.1&utm_medium=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-84241776-blog-122706847.pc_relevant_multi_platform_featuressortv2dupreplace&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-1-84241776-blog-122706847.pc_relevant_multi_platform_featuressortv2dupreplace&utm_relevant_index=1

2.1

https://zhuanlan.zhihu.com/p/44860862

2.2

https://blog.csdn.net/jiachang98/article/details/120966639?spm=1001.2101.3001.6650.10&utm_medium=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-10-120966639-blog-84241776.pc_relevant_multi_platform_whitelistv4&depth_1-utm_source=distribute.pc_relevant.none-task-blog-2%7Edefault%7ECTRLIST%7ERate-10-120966639-blog-84241776.pc_relevant_multi_platform_whitelistv4&utm_relevant_index=11#t3

https://www.cnblogs.com/Peyton-Li/p/9772281.html

2.3

https://zhuanlan.zhihu.com/p/151231495

https://blog.csdn.net/zhpfeng10/article/details/108977526

2.4

https://www.zhihu.com/question/323578684/answer/753474442

https://zhuanlan.zhihu.com/p/470026382

2.5

https://zhuanlan.zhihu.com/p/29846048

3.2

https://zhuanlan.zhihu.com/p/259999988

https://blog.csdn.net/zhangyufeikk/article/details/94594646

4.1

https://zhuanlan.zhihu.com/p/419854977

5.1

https://zhuanlan.zhihu.com/p/79061355

6.1

https://www.cnblogs.com/guoben/p/13063197.html

结尾

还有上次作业的没有补上，先提交一版吧

自我评价还是做的比较潦草，不过比上次应该大概好一点?

上次作业的提交时间是赶不上了，唉

这周被老师征用了三天，人裂开。。。