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. **在什么条件下，x 有解且唯一？**

   假设A 为n x n 的方阵，那么当b 为全是0， 维数是n的列向量，且A的秩为n 时， x有唯一解。 可以通过初等变换将A 变形。

2. **高斯消元法的原理是什么？**

   高斯消元法的算法如下：

   1. 构造增广矩阵，即系数矩阵A增加上常数向量b
   2. 通过以交换行、某行乘以非负常数和两行相加这三种初等变化将原系统转化为更简单的三角形式. 如
   3. 从而得到简化的三角方阵组 求解

3. **QR 分解的原理是什么？**

   QR 分解可以把一个m x n 矩阵分解成一个m x m半正交矩阵与一个m x n上三角矩阵的积。其中Q是正交矩阵 R 是上三角矩阵。可以用来解线性最小二乘法问题， 也可以用来解求特征值

4. **Cholesky 分解的原理是什么？**

   Cholesky 分解是正定的Hermite矩阵分解成一个下三角矩阵与其共轭转置之乘积的分解， 可以用来快速求解线性方程组。

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。你可以参考本次课 用到的 useEigen 例程。**

```
#include <iostream>

#include <ctime>
#include <Eigen/Core>
#include <Eigen/Cholesky>
#include <Eigen/Dense>
#include <cassert>
#define MATRIX_SIZE 100

using namespace std;
using namespace Eigen;

int main( int argc, char** argv )
{
    MatrixXd matrix_A = Eigen::MatrixXd::Random( MATRIX_SIZE, MATRIX_SIZE
);;
```

```
    Matrix< double, Eigen::Dynamic,  1> v_Nd;
    v_Nd = Eigen::MatrixXd::Random( MATRIX_SIZE,1 );
    matrix_A = matrix_A.transpose() * matrix_A +
MatrixXd::Identity(MATRIX_SIZE, MATRIX_SIZE);
    // Use QR
    clock_t time_stt = clock();
    Eigen::Matrix<double, Eigen::Dynamic, 1> qr_result =
matrix_A.colPivHouseholderQr().solve(v_Nd);
    cout <<"time use in Qr decomposition is " <<1000*  (clock() -
time_stt)/(double)CLOCKS_PER_SEC <<"ms" << endl;
    // Use Cholesky
    time_stt = clock();
    LLT<Matrix<double, 100, 100>> llt;
    MatrixXd cholesky_result = matrix_A.ldlt().solve(v_Nd);
    cout <<"time use in Choesky is " <<1000*  (clock() -
time_stt)/(double)CLOCKS_PER_SEC <<"ms" << endl;
    assert(qr_result.isApprox(cholesky_result));
    cout << "qr result equal to cholesky result" << endl;
    return 0;
}
```

运行截图：

```
/Users/jinxwu/CLionProjects/eigen/cmake-build-debug/eigen
time use in Qr decomposition is 16.396ms
time use in Choesky is 3.192ms
qr result equal to cholesky result

Process finished with exit code 0
```

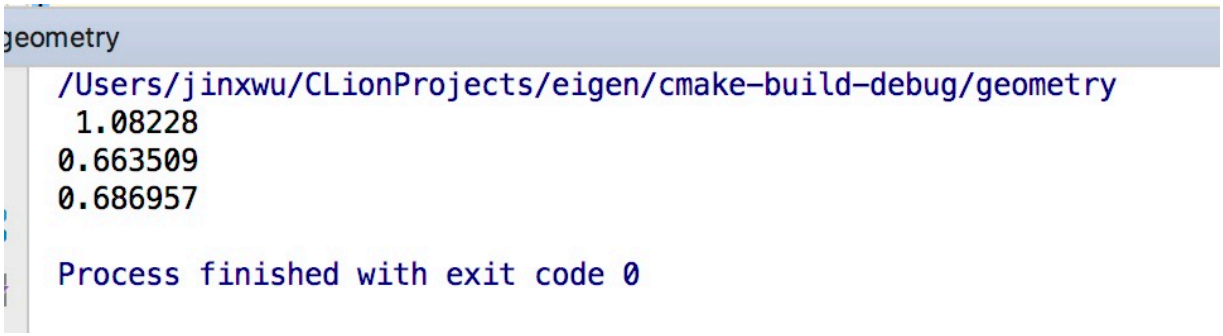
几何运算练习

```
#include <Eigen/Core>
#include <Eigen/Geometry>
#include <iostream>
#include <cassert>
using namespace Eigen;
using namespace std;

int main(){
    // q1 为小萝卜一号变换到世界坐标系的旋转，  将虚部取反
    Quaterniond q1 = Quaterniond(0.55, -0.3, -0.2, -0.2);
    Quaterniond q2 = Quaterniond(-0.1, 0.3, -0.7, 0.2);
    // 归一化
```

```
    q1.normalize();
    q2.normalize();
    // 平移
    Vector3d t1(0.7, 1.1, 0.2);
    Vector3d t2(-0.1, 0.4, 0.8);
    //目标点在小萝卜一号的坐标
    Vector3d p1 = Vector3d(0.5, -0.1, 0.2);
    //目标点在世界的坐标，先逆向平移，再旋转
    Vector3d real_pos;
    real_pos = -t1 + p1;
    real_pos = q1*real_pos;
    //目标点在小萝卜二号的坐标，先平移，再旋转
    Vector3d pos_2 = q2*real_pos + t2;
    cout << pos_2 << endl;
    assert(pos_2.isApprox(Vector3d(1.08228, 0.663509, 0.686957),
0.00001));
    return 0;
}
```

运行截图



旋转的表达

课程中提到了旋转可以用旋转矩阵、旋转向量与四元数表达，其中旋转矩阵与四元数是日常应用中常 见的表达方式。请根据课件知识，完成下述内容的证明

1. 设有旋转矩阵 R，证明 $\boldsymbol{R}^{\boldsymbol{T}}\boldsymbol{R}=\boldsymbol{I}$ 且$det\boldsymbol{R}=+1$

   答：依据书本的定义设$e_1, e_2, e_2, e_1', e_2', e_3'$ 为基底。

$$R = \begin{matrix} \mathbf{e}_1^T\mathbf{e}_1' & \mathbf{e}_1^T\mathbf{e}_2' & \mathbf{e}_1^T\mathbf{e}_3' \\ \mathbf{e}_2^T\mathbf{e}_1' & \mathbf{e}_2^T\mathbf{e}_2' & \mathbf{e}_2^T\mathbf{e}_1' \\ \mathbf{e}_3^T\mathbf{e}_1' & \mathbf{e}_3^T\mathbf{e}_2' & \mathbf{e}_3^T\mathbf{e}_3' \end{matrix}$$

$$R^T = \begin{matrix} \mathbf{e}_1^T\mathbf{e}_1' & \mathbf{e}_2^T\mathbf{e}_1' & \mathbf{e}_3^T\mathbf{e}_1' \\ \mathbf{e}_1^T\mathbf{e}_2' & \mathbf{e}_2^T\mathbf{e}_2' & \mathbf{e}_3^T\mathbf{e}_2' \\ \mathbf{e}_1^T\mathbf{e}_3' & \mathbf{e}_2^T\mathbf{e}_3' & \mathbf{e}_3^T\mathbf{e}_3' \end{matrix}$$

R的第一行乘R^T的第一列：$\mathbf{e}_1^T\mathbf{e}_1'\mathbf{e}_1^T\mathbf{e}_1' + \mathbf{e}_1^T\mathbf{e}_2'\mathbf{e}_2^T\mathbf{e}_1' + \mathbf{e}_1^T\mathbf{e}_3'\mathbf{e}_1^T\mathbf{e}_3' = 1+1+1 = 3$，因为是长度为1相同的向量点乘

R的第一行乘R^T的第二列：$\mathbf{e}_1^T\mathbf{e}_1'\mathbf{e}_2^T\mathbf{e}_1' + \mathbf{e}_1^T\mathbf{e}_2'\mathbf{e}_2^T\mathbf{e}_2' + \mathbf{e}_1^T\mathbf{e}_3'\mathbf{e}_2^T\mathbf{e}_3' = 0+0+0 = 0$。其中总有$\mathbf{e}_1^T\mathbf{e}_2^T = 0$，依此类推：

$$\boldsymbol{R^T R} = R^T = \begin{matrix} 3 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{matrix}$$

因为$\boldsymbol{R^T R} = \boldsymbol{I}$说明 $\boldsymbol{R}$ 实际上是一个正交矩阵， 而正交矩阵的秩为1， 由上式也显然可知.

---

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。

答：ε 的维度为3， η 的维度为1， 参见四元数的定义

---

3. 证明如下命题

定义运算 $^+$ 和 $^\oplus$ 为：

$$\boldsymbol{q}^+ = \begin{bmatrix} \eta\mathbf{1} + \boldsymbol{\varepsilon}^\times & \boldsymbol{\varepsilon} \\ -\boldsymbol{\varepsilon}^{\mathrm{T}} & \eta \end{bmatrix}, \quad \boldsymbol{q}^\oplus = \begin{bmatrix} \eta\mathbf{1} - \boldsymbol{\varepsilon}^\times & \boldsymbol{\varepsilon} \\ -\boldsymbol{\varepsilon}^{\mathrm{T}} & \eta \end{bmatrix}, \tag{1}$$

其中运算 $^\times$ 含义与 ^ 相同，即取 $\boldsymbol{\varepsilon}$ 的反对称矩阵（它们都成叉积的矩阵运算形式），$\mathbf{1}$ 为单位矩阵。请证明对任意单位四元数 $\boldsymbol{q}_1, \boldsymbol{q}_2$，四元数乘法可写成矩阵乘法：

$$\boldsymbol{q}_1 \cdot \boldsymbol{q}_2 = \boldsymbol{q}_1^+ \boldsymbol{q}_2 \tag{2}$$

或者

$$\boldsymbol{q}_1 \cdot \boldsymbol{q}_2 = \boldsymbol{q}_2^\oplus \boldsymbol{q}_1. \tag{3}$$

答：按照矩阵乘法的规则展开即可， 根据课本公式3.24

设$\mathbf{q_a} = [\varepsilon_1, \eta_1], \mathbf{q_b} = [\varepsilon_2, \eta_2]$。则$q_a q_b = [\eta_1\varepsilon_2 + \eta_2\varepsilon_1 + \varepsilon_1 \times \varepsilon_2, \eta_1\eta_2 - \varepsilon_1^T\varepsilon_2]^T$ (1

同时我们有：

$$
\begin{aligned}
&\begin{bmatrix} \eta_1 \mathbf{1}\varepsilon_1^{\times} & \varepsilon_1 \\ -\varepsilon_1^T & \eta_1 \end{bmatrix} \cdot [\varepsilon_2, \eta_2] \qquad (2 \\
&= \begin{bmatrix} (\eta_1 \mathbf{1} + \varepsilon_1^{\times})\varepsilon_2 + \varepsilon_1 \eta_2 \\ -\varepsilon_1^T \varepsilon_2 + \eta_1 \eta_2 \end{bmatrix} \\
&= \begin{bmatrix} \eta_1 \varepsilon_2 + \varepsilon_1 \times \varepsilon_2 + \varepsilon_1 \eta_2 \\ -\varepsilon_1^T \varepsilon_2 + \eta_1 \eta_2 \end{bmatrix}
\end{aligned}
$$

式(1) 与 (2) 显然相等，得证。

类似的， 对于$q_1 \cdot q_2 = q_2^{\oplus} q_1$ 的证明与上相同。

罗德里格斯公式的证明

罗德里格斯公式描述了从旋转向量到旋转矩阵的转换关系。设旋转向量长度为 θ，方向为 $\boldsymbol{n}$，那么旋转矩阵 $\boldsymbol{R}$ 为：

$$
\boldsymbol{R} = \cos\theta \boldsymbol{I} - (1 - \cos\theta)\boldsymbol{n}\boldsymbol{n}^T + \sin\theta \boldsymbol{n}^{\wedge}. \qquad (4)
$$

我们在课程中仅指出了该式成立，但没有给出证明。请你证明此式。

提示：参考https://en.wikipedia.org/wiki/Rodrigues%27_rotation_formula。

证：设 $\vec{v}$ 是某3维向量，其绕单位向量 $\vec{n}$ 旋转 θ 以后得到的向量为 $\vec{v}_{rot}$，接下来我们要证明：$\vec{v}_{rot} = R\vec{v}$ 且 $R = \cos\theta I + (1-\cos\theta)nn^T + \sin\theta n^\wedge$

(1) 对向量 $\vec{v}$，它可以分解成平行于单位向量 $\vec{n}$ 的向量 $\vec{v}_{\parallel}$ 与垂直于单位向量 $\vec{n}$ 的向量 $\vec{v}_{\perp}$ 之和：$\vec{v} = \vec{v}_{\parallel} + \vec{v}_{\perp}$，根据图 1

$$\vec{v}_{\parallel} = (\vec{v}\cdot\vec{n})\vec{n}$$

$$\vec{v}_{\perp} = -\vec{n}\times(\vec{n}\times\vec{v}) = \vec{v} - \vec{n}(\vec{n}\vec{v})$$

在 $\vec{v}$ 旋转过后，它的与单位向量 $\vec{n}$ 平行的部分不变，只有与其垂直的部分旋转 θ 度，因此，有：

$$\vec{v}_{\perp rot} = \cos\theta\,\vec{v}_{\perp} + \sin\theta\,\vec{n}\times\vec{v}$$

于是有

$$\vec{v}_{rot} = \vec{v}_{\perp rot} + \vec{v}_{\parallel rot}$$

$$= \vec{v}_{\parallel} + \cos\theta\,\vec{v}_{\perp} + \sin\theta\,\vec{n}\times\vec{v}$$

$$= \cos\theta\,\vec{v} + (1-\cos\theta)(\vec{n}\vec{v})\vec{n} + \sin\theta\,\vec{n}\times\vec{v}$$

把叉乘改成点乘，

$$= \cos\theta\,\vec{v} + (1-\cos\theta)\vec{n}n^T\vec{v} + \sin\theta\,\vec{n}^\wedge\vec{v}$$

$$= (\cos\theta I + (1-\cos\theta)\vec{n}\vec{n}^T + \sin\theta\,\vec{n}^\wedge)\vec{v}$$

得证。

四元数运算性质的验证

课程中介绍了单位四元数可以表达旋转。其中，在谈论用四元数 $\boldsymbol{q}$ 旋转点 $\boldsymbol{p}$ 时，结果为：

$$\boldsymbol{p}' = \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^{-1}. \tag{5}$$

我们说，此时 $\boldsymbol{p}'$ 必定为虚四元数（实部为零）。请你验证上述说法。

此外，上式亦可写成矩阵运算：$\boldsymbol{p}' = \boldsymbol{Q}\boldsymbol{p}$。请根据你的推导，给出矩阵 $\boldsymbol{Q}$。注意此时 $\boldsymbol{p}$ 和 $\boldsymbol{p}'$ 都是四元数形式的变量，所以 $\boldsymbol{Q}$ 为 4×4 的矩阵。

提示：如果使用第 4 题结果，那么有：

$$\begin{aligned} \boldsymbol{p}' &= \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^{-1} = \boldsymbol{q}^{+}\boldsymbol{p}^{+}\boldsymbol{q}^{-1} \\ &= \boldsymbol{q}^{+}\boldsymbol{q}^{-1^{\oplus}}\boldsymbol{v}. \end{aligned} \tag{6}$$

从而可以导出四元数至旋转矩阵的转换方式：

$$\boldsymbol{R} = \boldsymbol{q}^{+}\boldsymbol{q}^{-1^{\oplus}}. \tag{7}$$

答：假设$q = (a, \mathbf{n}), p = (0, \mathbf{v})$, 则 $p^{-1} = (a, -\mathbf{n})$, 依据四元数运算法则。

$$(a, \vec{n})(0, \vec{v})(a, -\vec{n})$$

$$= (0 - \vec{n}^T\vec{v},\ a\vec{v} + 0 + \vec{n}\times\vec{v})(a, -\vec{n})$$

$$= (-n^T\vec{v}a - (a\vec{v} + \vec{n}\times\vec{v})^T(-\vec{n}),\ \text{忽略})$$

$$= \left(-an^T\vec{v} + a\vec{v}^T\vec{n} + (\vec{n}\times\vec{v})^T\vec{n}\right)$$

由于实部可化为是 1×1 的向量，取转置

$$= \left[\underbrace{-a\vec{v}^T n + a\vec{n}^T v}_{(1)} + \underbrace{\vec{n}^T(\vec{n}\times\vec{v})}_{(2)}\right]$$

对于(1)，$\vec{n}$ 和 $\vec{v}$ 都是 (3×1) 的向量，设为 $\begin{pmatrix}v_1\\v_2\\v_3\end{pmatrix}, \begin{pmatrix}n_1\\n_2\\n_3\end{pmatrix}$ 则

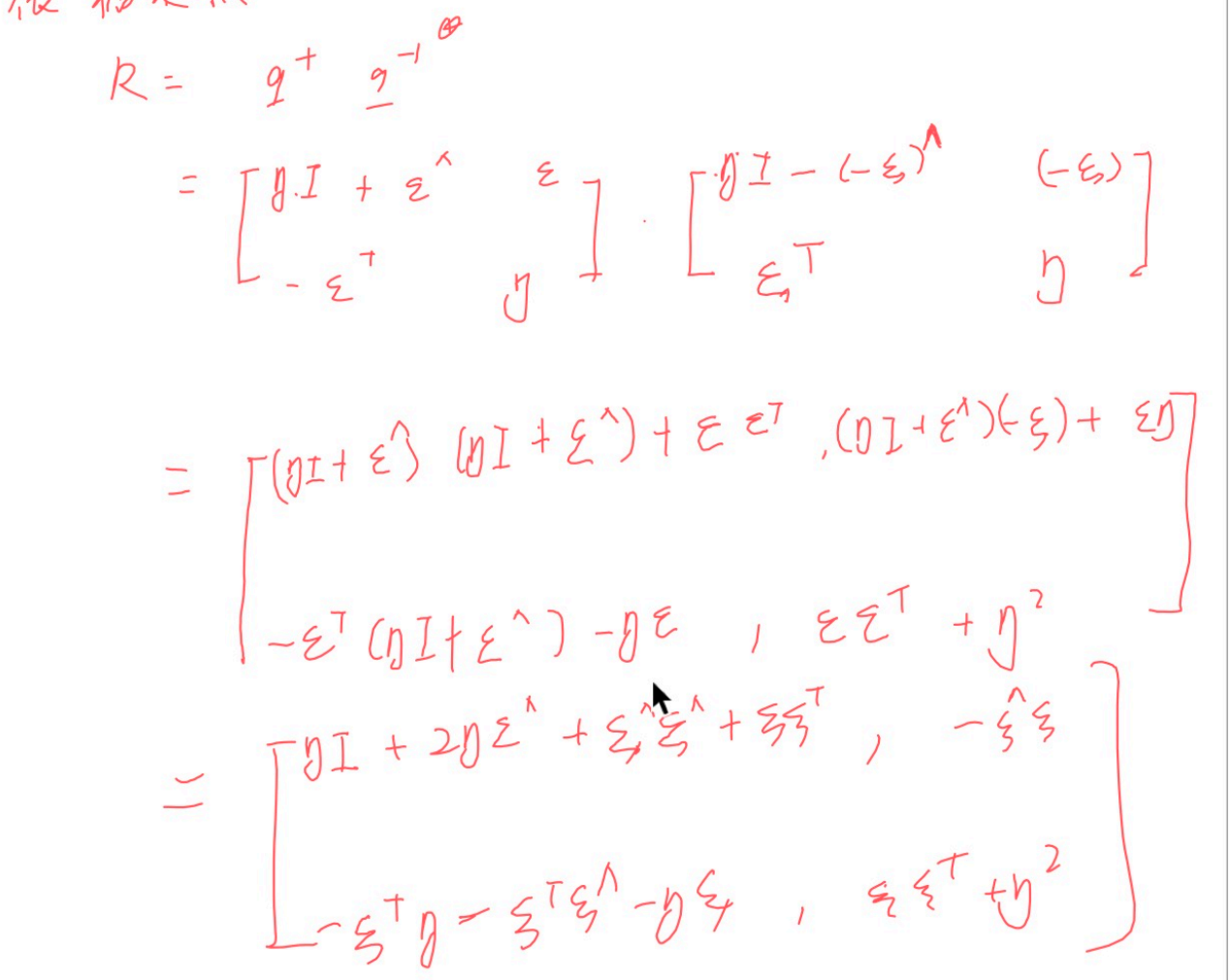
$\vec{v}^T\cdot\vec{n} = \vec{n}^T v = n_1v_1 + n_2v_2 + v_3v_3$ 故 式1 $=0$

对于(2) $\vec{n}^T(\vec{n}\times\vec{v}) = \underline{\vec{n}^T\cdot\hat{n}\ \vec{v}}$

$$= (n_1, n_2, n_3)\begin{pmatrix}0 & -n_3 & n_2\\ n_3 & 0 & -n_1\\ -n_2 & n_1 & 0\end{pmatrix}\vec{v}$$

矩阵乘法可知：

$= 0$



熟悉C++11

设有类 A，并有 A 类的一组对象，组成了一个 vector。现在希望对这个 vector 进行排序，但排序的方式由 A.index 成员大小定义。那么，在 C++11 的语法下，程序写成：

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

class A {
public:
    A(const int& i ) : index(i) {}
    int index = 0;
};

int main() {
    A a1(3), a2(5), a3(9);
    vector<A> avec{a1, a2, a3};
    std::sort(avec.begin(), avec.end(), [](const A&a1, const A&a2) {return a1.index<a2.index;});
    for ( auto& a: avec ) cout<<a.index<<" ";
    cout<<endl;
    return 0;
}
```

请说明该程序中哪些地方用到了 C++11 标准的内容。提示：请关注范围 for 循环、自动类型推导、lambda 表达式等内容。

利用了如下C++ 特性

1. auto 关键字， 从初始化表达式中推断出变量的数据类型。参见 line 17, `auto& a: avec`
2. for 循环, 简化的for循环, 不需要定义数组长度
3. 利用lambda 表达式来定义匿名的函数对象，参见line 16, `std::sort(avec.begin(), avec.end(), [](const A&a1, const A&a2) {return a1.index<a2.index;});` ， 这里sort 函数使用了匿名的lambda 函数来进行排序。