

0. 说明

本PDF文档为自动生成，如有遗漏的格式错误请及时告知！

1. 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:

问1-1：在什么条件下, x 有解且唯一？

A 为满秩矩阵，即 $r(A)=n$. (n 为方阵的阶数)。

问1-2：高斯消元法的原理是什么？

- 构造增广矩阵 $(A|b)$ ；
- 通过初等行变换，将矩阵化为阶梯阵（或三角式）；
- 最后通过回代求解；
- 其核心在于减少方程中的未知数个数，算法复杂度一般为 $O(n^3)$ 。

问1-3：QR 分解的原理是什么？

- 将矩阵分解为 Q 和 R 两部分，其中 Q 是标准正交矩阵， R 是一个上三角矩阵；
- 通过将矩阵 A 提前分解，可以降低解算线性方程的复杂度；
- 求解时一般先求出标准正交矩阵 Q ，然后就可以快速求出矩阵 R ；

$$A = QR \Rightarrow R = Q^{-1}A = Q^T A$$

- 常用方法有：Householder、Gram-Schmidt、Givens

问1-4：Cholesky 分解的原理是什么？

- 把矩阵分解为一个下三角矩阵以及它的共轭转置矩阵的乘积；
- 矩阵 A 为实对称正定矩阵；
- 通过将矩阵 A 提前分解，可以降低解算线性方程的复杂度；

问1-5：你需要使用 Eigen 库, 编写程序实现 A 为 100×100 随机矩阵时, 用 QR 和 Cholesky 分解求 x 的程序。

- 进行 Cholesky 分解需要矩阵 A 为实对称正定矩阵，代码中将生成的随机矩阵 M ，然后令 $A = M^T M$

下面进行非严格证明：

- 对称

$$A^T = (M^T M)^T = M^T M = A$$

- 正定

如果 $M^T M$ 存在负特征值，则二次多项式 $f(x) = x^T M^T M x$ 存在非零向量 x 使得 $f(x) < 0$

但是 $f(x) = \|Mx\|^2 \geq 0$, 所以矩阵 A 的特征值一定大于等于 0.

- 代码：

- useEigen.cpp

```
//求解Ax=b这个线性方程，其中A为100×100的随机方阵，b为向量
#include<iostream>
#include<ctime>

#include<eigen3/Eigen/Eigen>

using namespace std;
using namespace Eigen;

#define MATRIX_SIZE 100

int main(int argc, char** argv){
    srand((unsigned)time(NULL));
    MatrixXd M = MatrixXd::Random(MATRIX_SIZE, MATRIX_SIZE);
    MatrixXd b = VectorXd::Random(MATRIX_SIZE);
    //Cholesky分解要求矩阵A必须为实对称正交矩阵
    MatrixXd A = M.transpose()*M;
    //QR(Eigen提供了许多QR分解函数，这里选择colPivHouseholderQr)
    VectorXd x_qr = A.colPivHouseholderQr().solve(b);
    assert(b.isApprox(A*x_qr));
    cout<<"Here is a solution(QR decomposition) x to the equation
Ax=b:"<<endl<<x_qr.transpose()<<endl;
    //Cholesky
    VectorXd x_cholesky = A.llt().solve(b);
    assert(b.isApprox(A*x_cholesky));
    cout<<"Here is a solution(Cholesky decomposition) x to the equation
Ax=b:"<<endl<<x_cholesky.transpose()<<endl;

    return 0;
}
```

- CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.3)
SET(CMAKE_BUILD_TYPE "Release")
PROJECT (Chapters2)

INCLUDE_DIRECTORIES(${PROJECT_SOURCE_DIR}/include)
INCLUDE_DIRECTORIES("/usr/include/eigen3")

SET(SRC_LIST ${PROJECT_SOURCE_DIR}/src/useEigen.cpp)
ADD_EXECUTABLE(useEigen ${SRC_LIST})
```

- 运行结果

```

$ cd /Users/optimiflex/~/github/SLAH/Nonworks/Chapter2/build$ make
Scanning dependencies of target useEigen
[ 50%] Building CXX object useEigen.dir/src/useEigen.cpp.o
[100%] Linking CXX executable useEigen
$ cd /Users/optimiflex/~/github/SLAH/Nonworks/Chapter2/build$ ./useEigen
Here is a solution(QR decomposition) x to the equation Ax=b:
75.9122 -68.6861 -57.7638 27.9584 66.3489 -67.6842 68.4799 25.2611 -108.628 -31.4196 44.3947 -13.3684 136.295 -33.4168 -46.8077 -126.45 17.526 -128.093 16.7586 -74.8281 -59.7521 -5.35177 8.22191 -2.78754 -27.1836
-46.4666 -15.3383 61.7995 -44.1957 -2.81524 -26.3617 -16.1567 -46.4125 -65.8476 27.9878 2.9785 -75.827 23.8278 -17.1029 -29.8336 43.3377 25.3252 -54.3486 72.2674 19.5884 12.1235 -47.6983 -1.99269 66.934 11.848
34.4898 58.2888 42.164 -56.8642 -6.78499 7.63958 -7.4485 -47.8742 -66.9177 4.4644 -88.4871 24.5772 2.48859 10.4886 -138.467 -63.8546 59.9756 -76.1114 12.1841 71.9581 -26.3462 -122.112 -38.8945 87.6637 -2.768
65.024 83.5911 42.057 25.834 -28.8723 72.587 22.573 9.54515 5.51888 22.3615 -34.1302 37.7213 -22.7945 37.3858 24.5791 26.4259 -6.33814 -82.8678 17.5193 -26.2583 3.36853 -60.9979 19.6119 97.1411 -17.1
982
Here is a solution(Cholesky decomposition) x to the equation Ax=b:
75.9122 -68.6861 -57.7638 27.9584 66.3489 -67.6842 68.4799 25.2611 -108.628 -31.4196 44.3947 -13.3684 136.295 -33.4168 -46.8077 -126.45 17.526 -128.093 16.7586 -74.8281 -59.7521 -5.35177 8.22191 -2.78754 -27.1836
-46.4666 -15.3383 61.7995 -44.1957 -2.81524 -26.3617 -16.1567 -46.4125 -65.8476 27.9878 2.9785 -75.827 23.8278 -17.1029 -29.8336 43.3377 25.3252 -54.3486 72.2674 19.5884 12.1235 -47.6983 -1.99269 66.934 11.848
34.4898 58.2888 42.164 -56.8642 -6.78499 7.63958 -7.4485 -47.8742 -66.9177 4.4644 -88.4871 24.5772 2.48859 10.4886 -138.467 -63.8546 59.9756 -76.1114 12.1841 71.9581 -26.3462 -122.112 -38.8945 87.6637 -2.768
65.024 83.5911 42.057 25.834 -28.8723 72.587 22.573 9.54515 5.51888 22.3615 -34.1302 37.7213 -22.7945 37.3858 24.5791 26.4259 -6.33814 -82.8678 17.5193 -26.2583 3.36853 -60.9979 19.6119 97.1411 -17.1
982
Here is a solution(QR decomposition) x to the equation Ax=b:
15.4874 -4.89844 7.27164 17.8357 11.5276 2.37256 -18.9688 6.12575 -2.89254 5.85349 -15.0069 3.62883 -7.25949 -5.26671 -8.19817 0.559309 -1.51697 0.559473 0.939801 6.51413 0.468949 10.7174 -3.
17836 -8.916358 0.8234849 16.1445 -5.993 -4.38428 0.88968 -7.38111 2.93688 -1.51486 -8.716716 -6.6674 -5.62997 -18.9354 1.43687 9.12845 -3.25881 16.6195 22.2899 -4.28884 14.4792 -11.588
5.98633 2.74821 -8.44383 3.61865 5.98788 -9.92612 -18.8417 1.1985 0.6489527 16.1329 8.12926 -7.12455 18.8221 5.7325 17.1859 1.97953 -5.51324 8.83899 2.58732 -8.51486 0.939548 2.36796
13.5512 0.288485 -9.55697 12.7241 -3.14086 11.9258 -10.2489 17.6677 9.27905 -8.83389 -14.9143 -4.17468 25.6359 16.8927 25.3626 -11.6352 3.27162 -4.21885 0.455545 -0.46203 2.1225 -28.0159 11.
49 12.0731 2.95855 -14.5488 -2.28575 2.47883 -7.96215 -2.68106 0.29193 -28.8434 0.871896
Here is a solution(Cholesky decomposition) x to the equation Ax=b:
15.4874 -4.89844 7.27164 17.8357 11.5276 2.37256 -18.9688 6.12575 -2.89254 5.85349 -15.0069 3.62883 -7.25949 -5.26671 -8.19817 0.559309 -1.51697 0.559473 0.939801 6.51413 0.468949 10.7174 -3.
17836 -8.916358 0.8234849 16.1445 -5.993 -4.38428 0.88968 -7.38111 2.93688 -1.51486 -8.716716 -6.6674 -5.62997 -18.9354 1.43687 9.12845 -3.25881 16.6195 22.2899 -4.28884 14.4792 -11.588
5.98633 2.74821 -8.44383 3.61865 5.98788 -9.92612 -18.8417 1.1985 0.6489527 16.1329 8.12926 -7.12455 18.8221 5.7325 17.1859 1.97953 -5.51324 8.83899 2.58732 -8.51486 0.939548 2.36796
13.5512 0.288485 -9.55697 12.7241 -3.14086 11.9258 -10.2489 17.6677 9.27905 -8.83389 -14.9143 -4.17468 25.6359 16.8927 25.3626 -11.6352 3.27162 -4.21885 0.455545 -0.46203 2.1225 -28.0159 11.
49 12.0731 2.95855 -14.5488 -2.28575 2.47883 -7.96215 -2.68106 0.29193 -28.8434 0.871896
$ cd /Users/optimiflex/~/github/SLAH/Nonworks/Chapter2/build$

```

2. 几何运算练习

- useGeometry.cpp

```

#include<iostream>

#include<eigen3/Eigen/Eigen>

using namespace std;
using namespace Eigen;

int main(int argc, char** argv){
    Quaterniond q1={0.55,0.3,0.2,0.2},q2={-0.1,0.3,-0.7,0.2};
    Vector3d t1={0.7,1.1,0.2},t2={-0.1,0.4,0.8},p1={0.5,-0.1,0.2},p2;

    q1=q1.normalized();
    q2=q2.normalized();

    Isometry3d Tcw1(q1),Tcw2(q2);
    Tcw1.pretranslate(t1);
    Tcw2.pretranslate(t2);

    p2=Tcw2* Tcw1.inverse()*p1;
    cout<<"p2 = "<<p2.transpose()<<endl;
    return 0;
}

```

- CMakeLists.txt

```

cmake_minimum_required(VERSION 2.8.3)
SET(CMAKE_BUILD_TYPE "Release")
PROJECT (Chapters2)

add_compile_options(-std=c++11)

INCLUDE_DIRECTORIES(${PROJECT_SOURCE_DIR}/include)
INCLUDE_DIRECTORIES("/usr/include/eigen3")

SET(SRC_LIST ${PROJECT_SOURCE_DIR}/src/useEigen.cpp)
ADD_EXECUTABLE(useEigen ${SRC_LIST})

ADD_EXECUTABLE(useGeometry ${PROJECT_SOURCE_DIR}/src/useGeometry.cpp)

```

- 运行结果

```
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Homeworks/Chapter2/build$ cmake ..
-- Configuring done
-- Generating done
-- Build files have been written to: /home/iusl/lyq/github/SLAM/Homeworks/Chapter2/build
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Homeworks/Chapter2/build$ make
[ 25%] Building CXX object CMakeFiles/useGeometry.dir/src/useGeometry.cpp.o
[ 50%] Linking CXX executable useGeometry
[ 50%] Built target useGeometry
[ 75%] Building CXX object CMakeFiles/useEigen.dir/src/useEigen.cpp.o
[100%] Linking CXX executable useEigen
[100%] Built target useEigen
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Homeworks/Chapter2/build$ ./useGeometry
p2 = 1.08228 0.663509 0.686957
iusl@iusl-OptiPlex-7060:~/lyq/github/SLAM/Homeworks/Chapter2/build$
```

3. 旋转的表达

证3-1：设有旋转矩阵 R , 证明 $R^T R = I$ 且 $\det R = +1$

旋转矩阵 R_B^A 可以表示为：

$$R_B^A = [\hat{X}_B^A \quad \hat{Y}_B^A \quad \hat{Z}_B^A] = \begin{bmatrix} \hat{X}_B \cdot \hat{X}_A & \hat{Y}_B \cdot \hat{X}_A & \hat{Z}_B \cdot \hat{X}_A \\ \hat{X}_B \cdot \hat{Y}_A & \hat{Y}_B \cdot \hat{Y}_A & \hat{Z}_B \cdot \hat{Y}_A \\ \hat{X}_B \cdot \hat{Z}_A & \hat{Y}_B \cdot \hat{Z}_A & \hat{Z}_B \cdot \hat{Z}_A \end{bmatrix} \dots\dots\dots (3-1)$$

观察式子 (3-1) 可得：

$$R_B^A = [\hat{X}_B^A \quad \hat{Y}_B^A \quad \hat{Z}_B^A] = \begin{bmatrix} \hat{X}_A^{BT} \\ \hat{Y}_A^{BT} \\ \hat{Z}_A^{BT} \end{bmatrix} = R_A^{BT} \dots\dots\dots (3-2)$$

从式子 (3-2) 可得：

$$R_B^A R_A^B = R_A^{BT} R_A^B = I \Rightarrow \text{旋转矩阵为正交矩阵}$$

因为旋转矩阵是正交矩阵，所以其行列式等于1或者-1，行列式的符号反映了基底的定向变化，因为我们一直使用右手法则，所以基底方向不变，取正号。

如果从几何上来考虑，旋转变换未改变其各向量之间的夹角和模的大小，从而向量构成的“体积”也没变，所以 $\det R = +1$ 。

证3-2：设有四元数 q , 我们把虚部记为 ϵ , 实部记为 η , 那么 $q = (\epsilon, \eta)$ 。请说明 ϵ 和 η 的维度

四元数都是由实数加上三个虚数单位 i, j, k 组成，一般可表示为 $q = a + bi + cj + dk$ ，即 $\eta = a, \epsilon = [b, c, d]^T$ 。所以实部的维度为1, 虚部的维度为3。

证3-3：定义运算⁺和[⊕]为：

$$q^+ = \begin{bmatrix} \eta \mathbf{I} + \epsilon^\times & \epsilon \\ -\epsilon^T & \eta \end{bmatrix}, q^\oplus = \begin{bmatrix} \eta \mathbf{I} - \epsilon^\times & \epsilon \\ -\epsilon^T & \eta \end{bmatrix}$$

其中运算 \times 含义与 \wedge 相同，即取 ϵ 的反对称矩阵， \mathbf{I} 为单位矩阵，请证明对任意单位四元数 q_1, q_2 ，四元数乘法可以写成矩阵乘法：

$$q_1 q_2 = q_1^+ q_2 \quad \text{或} \quad q_1 q_2 = q_2^\oplus q_1$$

令 $q_1 = bi + cj + dk + a$, $q_2 = fi + gj + hk + e$, 则 $\eta_1 = a, \eta_2 = e; \epsilon_1 = [b, c, d]^T, \epsilon_2 = [f, g, h]^T$
:

$$\begin{aligned} q_1 q_2 &= (bi + cj + dk + a)(fi + gj + hk + e) = \\ &\quad (ae - bf - cg - dh) + \\ &\quad (be + af - dg + ch)i \\ &\quad (ce + df + ag - bh)j \\ &\quad (de - cf + bg + ah)k \\ &= \\ &\begin{bmatrix} a & -d & c & b \\ d & a & -b & c \\ -c & b & a & d \\ -b & -c & -d & a \end{bmatrix} \begin{bmatrix} f \\ g \\ h \\ e \end{bmatrix} = \begin{bmatrix} \eta \mathbf{I} + \epsilon^\times & \epsilon \\ -\epsilon^T & \eta \end{bmatrix} q_2 = q_1^+ q_2 \end{aligned}$$

同理可证： $q_1 q_2 = q_2^\oplus q_1$

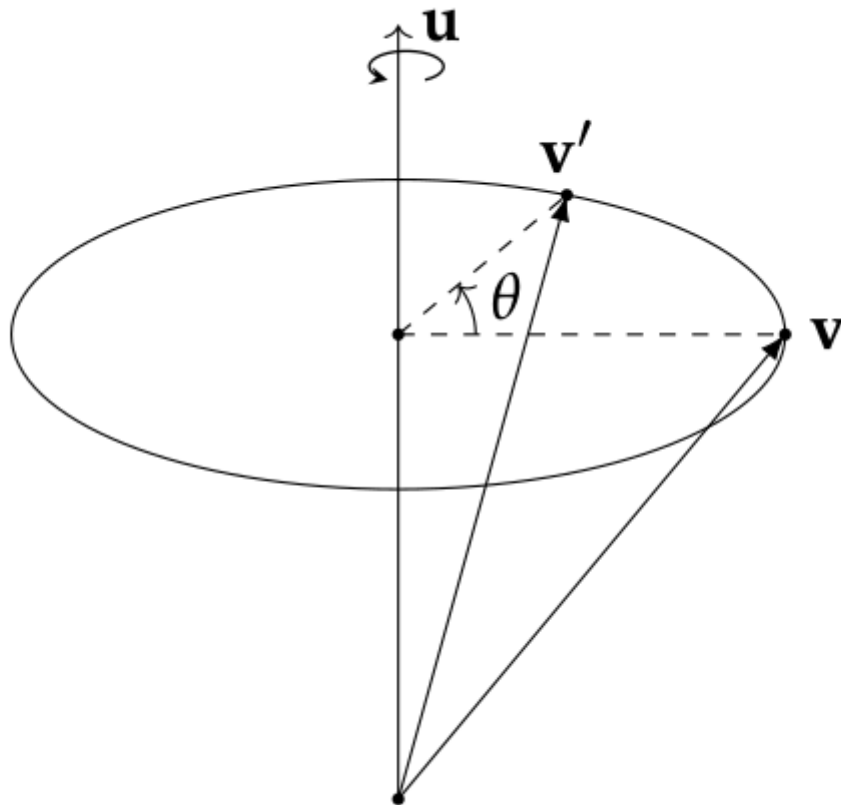
4. 罗德里格斯公式的证明

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 \mathbf{n} , 那么旋转矩阵 R 为:

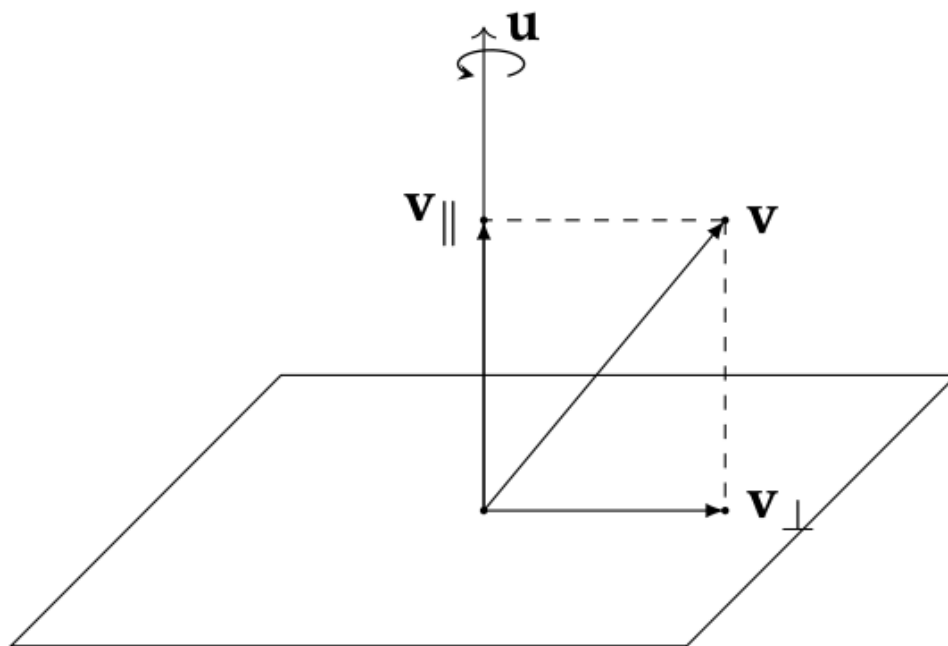
$$R = \cos\theta \mathbf{I} + (1 - \cos\theta) \mathbf{n} \mathbf{n}^T + \sin\theta \mathbf{n}^\wedge.$$

证4-1：证明罗德里格斯公式

(这里证明参考了<https://github.com/Krasjet/quaternion>)



上图表示的意思是向量 \mathbf{v} 绕旋转轴 \mathbf{u} （单位向量）旋转 θ 角度后变为 \mathbf{v}' 。



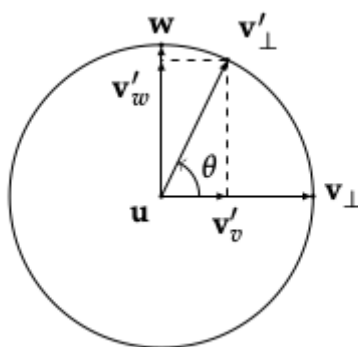
将向量 \mathbf{v} 分解为分别平行和垂直旋转轴的两个向量，即 $\mathbf{v} = \mathbf{v}_{\parallel} + \mathbf{v}_{\perp}$ ；从而 $\mathbf{v}' = \mathbf{v}'_{\parallel} + \mathbf{v}'_{\perp}$ 。

其中 $\mathbf{v}_{\parallel} = (\mathbf{u} \cdot \mathbf{v})\mathbf{u}$, $\mathbf{v}_{\perp} = \mathbf{v} - \mathbf{v}_{\parallel} = \mathbf{v} - (\mathbf{u} \cdot \mathbf{v})\mathbf{u}$ 。

- \mathbf{v}_{\parallel} 的旋转

因为其平行于且重合于旋转轴，所以旋转前后不变，即 $\mathbf{v}'_{\parallel} = \mathbf{v}_{\parallel}$ 。

- \mathbf{v}_{\perp} 的旋转



\mathbf{v}_{\perp} 的旋转位于一个平面内，同时为了方便从数学上描述这个旋转过程，构造一个向量 $\mathbf{w} = \mathbf{u} \times \mathbf{v}_{\perp}$ ，因为 $\|\mathbf{u}\| = 1$ ，可得 $\|\mathbf{w}\| = \|\mathbf{v}_{\perp}\|$ ，从而：

$$\mathbf{v}'_{\perp} = \mathbf{v}'_v + \mathbf{v}'_w = \cos(\theta)\mathbf{v}_{\perp} + \sin(\theta)\mathbf{w} = \cos(\theta)\mathbf{v}_{\perp} + \sin(\theta)(\mathbf{u} \times \mathbf{v}_{\perp})$$

综上所述可得：

$$\mathbf{v}' = \mathbf{v}'_{\parallel} + \mathbf{v}'_{\perp} = \mathbf{v}_{\parallel} + \cos(\theta)\mathbf{v}_{\perp} + \sin(\theta)(\mathbf{u} \times \mathbf{v}_{\perp})$$

又因为 $\mathbf{u} \times \mathbf{v}_{\perp} = \mathbf{u} \times (\mathbf{v} - \mathbf{v}_{\parallel}) = \mathbf{u} \times \mathbf{v}$ ，最后可得：

$$\begin{aligned}
\mathbf{v}' &= \mathbf{v}_{\parallel} + \cos(\theta)\mathbf{v}_{\perp} + \sin(\theta)(\mathbf{u} \times \mathbf{v}_{\perp}) \\
&= (\mathbf{u} \cdot \mathbf{v})\mathbf{u} + \cos(\theta)(\mathbf{v} - (\mathbf{u} \cdot \mathbf{v})\mathbf{u}) + \sin(\theta)(\mathbf{u} \times \mathbf{v}) \\
&= \cos(\theta)\mathbf{v} + (1 - \cos(\theta))(\mathbf{u} \cdot \mathbf{v})\mathbf{u} + \sin(\theta)(\mathbf{u} \times \mathbf{v}) \\
&= [\cos(\theta)\mathbf{I} + (1 - \cos(\theta))\mathbf{u}\mathbf{u}^T + \sin(\theta)\mathbf{u}^{\wedge}]\mathbf{v}
\end{aligned}$$

(这里用到了正交投影公式和叉乘的矩阵运算形式)

在本题中, 旋转轴 $\mathbf{u} = \mathbf{n}$, 故得到 $R = \cos\theta\mathbf{I} + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta\mathbf{n}^{\wedge}$

证4-2：利用罗德里格斯公式证明 $R^{-1} = R^T$

通过直接计算 $RR^T = I$ 来证明：

```
syms x y z theta real;

n = [x,y,z]';
R = cos(theta)*eye(3)+(1-cos(theta))*n*n'+sin(theta)*skew(n);
```

通过计算可得：

$$RR^T = \begin{bmatrix} ((1-c\theta)x^2+c\theta)^2+(zs\theta+xy(c\theta-1))^2+(ys\theta-xz(c\theta-1))^2 & xy(c\theta-1)^2(x^2+y^2+z^2-1) & xz(c\theta-1)^2(x^2+y^2+z^2-1) \\ xy(c\theta-1)^2(x^2+y^2+z^2-1) & ((1-c\theta)y^2+c\theta)^2+(xs\theta+yz(c\theta-1))^2+(zs\theta-xy(c\theta-1))^2 & yz(c\theta-1)^2(x^2+y^2+z^2-1) \\ xz(c\theta-1)^2(x^2+y^2+z^2-1) & yz(c\theta-1)^2(x^2+y^2+z^2-1) & ((1-c\theta)z^2+c\theta)^2+(ys\theta+xz(c\theta-1))^2+(xs\theta-yz(c\theta-1))^2 \end{bmatrix}$$

代入 $x^2 + y^2 + z^2 = 1$ 可得 $RR^T = I \Rightarrow R^{-1} = R^T$ 。

5. 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 q 旋转点 p 时,结果为:

$$p' = qpq^{-1}$$

证5-1：p' 必定为虚四元数(实部为零)

这里的点 p 写成四元数形式分别为 $[0, \mathbf{x}, \mathbf{y}, \mathbf{z}]$, 即 $[0, \mathbf{v}]$

$$p' = qpq^{-1} = q[0, \mathbf{v}]q^{-1}$$

令 $q = [\cos(\theta/2), \sin(\theta/2)\mathbf{u}]$, 则：

$$p' = [\cos(\theta/2), \sin(\theta/2)\mathbf{u}][0, \mathbf{v}][\cos(\theta/2), -\sin(\theta/2)\mathbf{u}]$$

根据Graßmann 积的计算公式可得实部为：

$$\begin{aligned}
& -\sin(\theta/2)\cos(\theta/2)\mathbf{u} \cdot \mathbf{v} - (\cos(\theta/2)\mathbf{v} + \sin(\theta/2)\mathbf{u} \times \mathbf{v}) \cdot (-\sin(\theta/2)\mathbf{u}) \\
& = -\sin(\theta/2)\cos(\theta/2)\mathbf{u} \cdot \mathbf{v} + \sin(\theta/2)\cos(\theta/2)\mathbf{u} \cdot \mathbf{v} + \sin^2(\theta/2)(\mathbf{u} \times \mathbf{v}) \cdot \mathbf{u} = 0
\end{aligned}$$

故可证 p' 必定为虚四元数。

扩：四元数与罗德里格斯公式的关系如下：

$$\mathbf{v}' = q\mathbf{v}q^* = q\mathbf{v}q^{-1} = qq^*\mathbf{v}_{\parallel} + qq\mathbf{v}_{\perp} = \mathbf{v}_{\parallel} + q^2\mathbf{v}_{\perp}$$

从而可得：

$$\begin{aligned}
\mathbf{v}' &= \mathbf{v}_{\parallel} + q^2\mathbf{v}_{\perp} = [0, (\mathbf{u} \cdot \mathbf{v})\mathbf{u}] + [0, \cos\theta\mathbf{v}_{\perp} + \sin\theta\mathbf{u} \times \mathbf{v}_{\perp}] \\
&= [0, \cos(\theta)\mathbf{v} + (1 - \cos(\theta))(\mathbf{u} \cdot \mathbf{v})\mathbf{u} + \sin(\theta)(\mathbf{u} \times \mathbf{v})]
\end{aligned}$$

问5-2：上式亦可写成矩阵运算： $p' = Qp$ 。请根据你的推导,给出矩阵 Q 。注意此时 p 和 p' 都是四元数形式的变量,所以 Q 为 4×4 的矩阵。

令四元数 $q = a + bi + cj + dk$, (注意这里与第四题不同, 实部在前, 最后推出来的结果也不一样)。

- 左乘这个四元数等价于左乘下面这个矩阵：

$$L(q) = \begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix}$$

- 右乘这个四元数等价于左乘下面这个矩阵：

$$R(q) = \begin{bmatrix} a & -b & -c & -d \\ b & a & d & -c \\ c & -d & a & b \\ d & c & -b & a \end{bmatrix}$$

则 $p' = qpq^{-1} = qpq^* = L(q)R(q)p = L(q)R(q)^T p$, 将 $a^2 + b^2 + c^2 + d^2 = 1$ 代入计算结果并化简可得：

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 - 2c^2 - 2d^2 & 2bc - 2ad & 2ac + 2bd \\ 0 & 2bc + 2ad & 1 - 2b^2 - 2d^2 & 2cd - 2ab \\ 0 & 2bd - 2ac & 2ab + 2cd & 1 - 2b^2 - 2c^2 \end{bmatrix}$$

6. 熟悉 C++11

```
#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。

- 用到了新的关键字 auto ；
- 用到了序列 for 循环 ；

- Lambda表达式；
 - 初始化列表；
-