

第二节课习题

高翔

2018 年 3 月 2 日

1 习题说明

- 第 i 节课习题所有材料打包在 $Li.zip$ 中, $\forall i = 1 \dots 8$ 。
- 习题分为若干种: **计算类**习题, 需要读者编程计算一个实际问题, 我们会附有参考答案以供自测。**操作类**习题, 会指导读者做一个具体的实验, 给出中间步骤截图或结果。**简述类**习题则提供阅读材料, 需要读者阅读材料后, 回答若干问题。
- 每个习题会有一定的分值。每次习题分值加和为 10 分。你需要获得 8 分以上才能得到“通过”的评价。带 * 的习题为附加题, 会在总分之外再提供一定的分值, 所以总和可能超过 10 分。换句话说, 你也可以选择一道附加题, 跳过一道正常题。
- 每道习题的给分由助教评判, 简述类习题可能存在一定开放性, 所以评分也存在主观因素。
- 请利用深蓝学院系统提交习题。每次习题我们会记通过与否。提交形式为 word 或 pdf 格式报告, 如有编程习题请提交可编译的源码。
- 为方便读者, 我通常会准备一些阅读材料, 放在 books/或 papers/目录下。请读者按个人需求使用这些材料。它们多数是从网络下载的, 如果侵犯到你的权利, 请及时告诉我。
- 每个习题会标注大致用时, 但视同学个人水平可能会有出入。
- 习题的完成情况会影响你对本课程内容的掌握程度, 请认真、独立完成。**习题总得分较高的同学将获得推荐资格。**

2 熟悉 Eigen 矩阵运算 (3 分, 约 2 小时)

Eigen (<http://eigen.tuxfamily.org>) 是常用的 C++ 矩阵运算库，具有很高的运算效率。大部分需要在 C++ 中使用矩阵运算的库，都会选用 Eigen 作为基本代数库，例如 Google Tensorflow, Google Ceres, GTSAM 等。本次习题，你需要使用 Eigen 库，编写程序，求解一个线性方程组。为此，你需要先了解一些有关线性方程组数值解法的原理。

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？
2. 高斯消元法的原理是什么？
3. QR 分解的原理是什么？
4. Cholesky 分解的原理是什么？
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。

提示：你可能需要参考相关的数学书籍或文章。请善用搜索引擎。Eigen 固定大小矩阵最大支持到 50，所以你会用到动态大小的矩阵。

m : 行向量数目, n : 列向量数目.

①, $Ax = b$ 有解: $\text{rank}(A) = \text{rank}(A, b)$.

矩阵 A 的列向量中的线性无关的列向量数目和其增广矩阵的线性无关的列向量数目相等，则意味着列向量 b 可以被 A 中的列向量而线性组合表示。

$Ax = b$ 有唯一解: $\text{rank}(A) = \text{rank}(A, b) = n$. 即矩阵 A 中的线性无关的列向量数目 = n . 列主元个数为 n . 没有无关变量，则必有唯一解。

② 将方程组中的一方程的未知数用含有另一未知数的代数式表示，并将其代入到另一方程中，这就消去了一未知数，得到一解。

Eg. 我们 $A \in \mathbb{R}^{3 \times 3}$ 找 A^{-1} .

那么
$$\left[A \mid \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \right] \rightarrow \left[\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \mid A^{-1} \right]$$

↑ 所求结果.

③ QR 分解原理.

QR 分解就是将矩阵分解成一个正交矩阵与一个上三角矩阵的积.

$$m \begin{bmatrix} A \\ n \end{bmatrix} = m \begin{bmatrix} Q \\ m \end{bmatrix} \begin{bmatrix} R \\ n \end{bmatrix} \rightarrow m \times n \text{ 矩阵} = \text{正交} \times \text{上三角}.$$

这里对于对称矩阵和非对称矩阵都适用.

Eg QR 分解

$$\begin{bmatrix} 0 & 3 & 1 \\ 0 & 4 & -2 \\ 2 & 1 & 2 \end{bmatrix}$$

Step 1: let $x_1 = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$ $x_2 = \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix}$ $x_3 = \begin{bmatrix} 1 \\ -2 \\ 2 \end{bmatrix}$

Step 2: 将 x_1, x_2, x_3 正交化:

$$y_1 = x_1 = \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

$$y_2 = x_2 - \frac{(x_2, y_1)}{(y_1, y_1)} y_1 = \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} - \frac{[3 \ 4 \ 1]^T [0 \ 0 \ 2]}{[0 \ 0 \ 2]^T [0 \ 0 \ 2]} \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

dot product.

$$= \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix}$$

$$y_3 = x_3 - \frac{(x_3, y_1)}{(y_1, y_1)} y_1 - \frac{(x_3, y_2)}{(y_2, y_2)} y_2$$

$$= \begin{bmatrix} 1 \\ -2 \\ 2 \end{bmatrix} - \frac{[1 \ -2 \ 2]^T [0 \ 0 \ 2]}{[0 \ 0 \ 2]^T [0 \ 0 \ 2]} \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} - \frac{[1 \ -2 \ 2]^T [3 \ 4 \ 0]}{[3 \ 4 \ 0]^T [3 \ 4 \ 0]} \begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ -2 \\ 2 \end{bmatrix} - 1 \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} + \frac{1}{5} \begin{bmatrix} 3 \\ 4 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 8/5 \\ -6/5 \\ 0 \end{bmatrix}$$

Step 3: Normalization (单位化)

$$e_1 = \frac{1}{\sqrt{2}} y_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad e_2 = \frac{1}{\sqrt{5}} y_2 = \begin{bmatrix} 3/\sqrt{5} \\ 4/\sqrt{5} \\ 0 \end{bmatrix}, \quad e_3 = \frac{1}{\sqrt{5}} y_3 = \begin{bmatrix} 4/\sqrt{5} \\ -3/\sqrt{5} \\ 0 \end{bmatrix}$$

Step 4: 从上面我们得知:

$$\begin{cases} y_1 = x_1 \\ y_2 = x_2 - \frac{1}{2}y_1 \\ y_3 = x_3 - y_1 + \frac{1}{2}y_2 \end{cases} \Rightarrow \begin{cases} x_1 = y_1 = 2e_1 \\ x_2 = \frac{1}{2}y_1 + y_2 = e_1 + 5e_2 \\ x_3 = y_1 - \frac{1}{2}y_2 + y_3 = 2e_1 - e_2 + 2e_3 \end{cases}$$

Step 5: 写成 QR 的形式:

$$Q = [e_1 \ e_2 \ e_3] = \begin{bmatrix} 0 & \frac{3}{5} & \frac{4}{5} \\ 0 & \frac{4}{5} & -\frac{3}{5} \\ 1 & 0 & 0 \end{bmatrix}$$

$$R = \overrightarrow{\text{看}} [x_1 \ x_2 \ x_3]^T = \begin{bmatrix} 2 & 1 & 2 \\ 0 & 5 & -1 \\ 0 & 0 & 2 \end{bmatrix}$$



④ Cholesky 分解原理:

Cholesky 分适用于 Hermitian positive-definite matrix.

Cholesky 分解即目标是将 A 变成: $A = LL^T$, L 为下三角矩阵.

$$A = \begin{bmatrix} a_{11} & A_{12}^T \\ A_{21} & A_{22} \end{bmatrix} \quad A_{11}, A_{11}^T \text{ 都是列向量. } A_{22} \text{ 是 } (n-1) \times (n-1) \text{ 的矩阵.}$$

$$\text{假设有 } L: L = \begin{bmatrix} l_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix}, \quad L^T = \begin{bmatrix} l_{11} & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix}$$

如果 $A = LL^T$. 那么:

$$\begin{bmatrix} a_{11} & A_{12}^T \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} l_{11} & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix} = \begin{bmatrix} l_{11}^2 & l_{11}L_{21}^T \\ L_{21}l_{11} & L_{21}L_{21}^T + L_{22}L_{22}^T \end{bmatrix}$$

$$\text{这里: } l_{11} = \sqrt{a_{11}} \quad L_{21} = \frac{1}{l_{11}} A_{21} \quad L_{22} \cdot L_{22}^T = A_{22} - L_{21}L_{21}^T$$

这样 $A_{22} - L_{21}L_{21}^T$ 可以求出来. 设为 A_{22}'

$$\text{那么 } L_{22}L_{22}^T = A_{22}' \rightarrow \text{另一个 Cholesky 分解问题}$$

\rightarrow Cholesky 具有递归性质.

$$\underline{\text{Ex}} \quad A = \begin{bmatrix} 25 & 15 & -5 \\ 15 & 18 & 0 \\ -5 & 0 & 11 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

$$l_{11} = \sqrt{a_{11}} = 5$$

$$L_{21} = \frac{1}{l_{11}} A_{21} = \frac{1}{5} \begin{bmatrix} 15 \\ -5 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

$$\left\{ \begin{array}{l} l_{11} = 5 \\ l_{21} = 3 \\ l_{31} = -1 \end{array} \right.$$

$$A_{22}^{-1} = A_{22} - L_{21} L_{21}^T = \begin{bmatrix} 18 & 0 \\ 0 & 11 \end{bmatrix} - \begin{bmatrix} 3 \\ -1 \end{bmatrix} \begin{bmatrix} 3 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 18 & 0 \\ 0 & 11 \end{bmatrix} - \begin{bmatrix} 9 & -3 \\ -3 & 1 \end{bmatrix} = \begin{bmatrix} 9 & 3 \\ 3 & 10 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 9 & 3 \\ 3 & 10 \end{bmatrix} = L_{22} \cdot L_{22}^T = \begin{bmatrix} l_{22} & 0 \\ l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{22} & l_{32} \\ 0 & l_{33} \end{bmatrix}$$

这里又重新是一下可以用 Cholesky 分解的式子。

重新定义这个 Cholesky 分解：

$$A = LL^T \quad \begin{bmatrix} 9 & 3 \\ 3 & 10 \end{bmatrix} = \begin{bmatrix} a_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} \\ 0 & l_{22} \end{bmatrix}$$

$$\Rightarrow l_{11} = \sqrt{a_{11}} = 3$$

$$l_{21} = \frac{1}{l_{11}} A_{21} = \frac{1}{3} \cdot 3 = 1$$

$$L_{22} \cdot L_{22}^T = A_{22} - L_{21} L_{21}^T \Rightarrow l_{22}^2 = 10 - 1^2 = 9 \Rightarrow l_{22} = 3.$$

$$\Rightarrow \text{第 } 2 \text{ 行第 } 1 \text{ 列元素为 } 3: \quad l_{11} = 3, \quad l_{21} = 1, \quad l_{22} = 3$$

那么和刚开始得出的第 1 行第 1 列元素在一起来，构成一个矩阵。

$$A = \begin{bmatrix} 5 & 0 & 0 \\ 3 & 3 & 0 \\ -1 & 1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 5 & 3 & -1 \\ 0 & 3 & 1 \\ 0 & 0 & 3 \end{bmatrix}$$

⑤ 用 Eigen 库实现 Matrix 的 QR 分解和 Cholesky 分解。

```
CAI Study > ch5 > hw_2_5 > C - prob25.cpp > ...
1 #include <iostream>
2 using namespace std;
3
4 #include<ctime>
5 #include<Eigen/Core>
6 #include<Eigen/Dense>
7 using namespace Eigen;
8
9 #define MATRIX_SIZE 100
10
11 int main(int argc, char **argv) {
12     Matrix<double, Dynamic, Dynamic> matrix_100;
13
14     matrix_100.resize(100, 100);
15     matrix_100 = MatrixXd::Random(100,100);
16     matrix_100 = matrix_100 * matrix_100.transpose();
17     Matrix<double, 100, 1> v_Nd = MatrixXd::Random(100,1);
18
19     // QR Decomposition
20     Matrix<double, 100, 1> x = matrix_100.colPivHouseholderQr().solve(v_Nd);
21     cout << "by QR, x = " << x.transpose() << endl;
22
23     // Cholesky Decomposition
24     x = matrix_100.ldlt().solve(v_Nd);
25     cout << "by Cholesky, x = " << x.transpose() << endl;
26
27     return 0;
28 }
29
```

3 几何运算练习 (2 分, 约 1 小时)

下面我们来练习如何使用 Eigen/Geometry 计算一个具体的例子。

设有小萝卜¹一号和小萝卜二号位于世界坐标系中。小萝卜一号的位姿为: $\mathbf{q}_1 = [0.55, 0.3, 0.2, 0.2]$, $\mathbf{t}_1 = [0.7, 1.1, 0.2]^T$ (\mathbf{q} 的第一项为实部)。这里的 \mathbf{q} 和 \mathbf{t} 表达的是 \mathbf{T}_{cw} , 也就是世界到相机的变换关系。小萝卜二号的位姿为 $\mathbf{q}_2 = [-0.1, 0.3, -0.7, 0.2]$, $\mathbf{t}_2 = [-0.1, 0.4, 0.8]^T$ 。现在, 小萝卜一号看到某个点在自身的坐标系下, 坐标为 $\mathbf{p}_1 = [0.5, -0.1, 0.2]^T$, 求该向量在小萝卜二号坐标系下的坐标。请编程实现此事, 并提交你的程序。

提示:

1. 四元数在使用前需要归一化。
2. 请注意 Eigen 在使用四元数时的虚部和实部顺序。
3. 参考答案为 $\mathbf{p}_2 = [1.08228, 0.663509, 0.686957]^T$ 。你可以用它验证程序是否正确。

```
1 #include <iostream>
2 using namespace std;
3
4 #include<Eigen/Core>
5 #include<Eigen/Dense>
6 using namespace Eigen;
7
8 int main(int argc, char** argv){
9     Quaterniond q1(0.55, 0.3, 0.2, 0.2);
10    Quaterniond q2(-0.1, 0.3, -0.7, 0.2);
11    q1.normalize();
12    q2.normalize();
13    Vector3d t1(0.7, 1.1, 0.2);
14    Vector3d t2(-0.1, 0.4, 0.8);
15    Vector3d p1(0.5, -0.1, 0.2);
16
17    Isometry3d T1w(q1);
18    Isometry3d T2w(q2);
19    T1w.pretranslate(t1);
20    T2w.pretranslate(t2);
21
22    Vector3d p2 = T2w * T1w.inverse() * p1;
23    cout << "p2 is: " << endl << p2.transpose() << endl;
24    return 0;
25 }
```

¹ 此处小萝卜指代机器人。

4 旋转的表达 (2 分, 约 1 小时)

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 \mathbf{R} ，证明 $\mathbf{R}^T \mathbf{R} = \mathbf{I}$ 且 $\det \mathbf{R} = +1^2$ 。
2. 设有四元数 \mathbf{q} ，我们把虚部记为 ε ，实部记为 η ，那么 $\mathbf{q} = (\varepsilon, \eta)$ 。请说明 ε 和 η 的维度。
3. 定义运算 $+$ 和 \oplus 为：

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

请证明对任意单位四元数 $\mathbf{q}_1, \mathbf{q}_2$ ，四元数乘法可写成矩阵乘法：

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = \mathbf{q}_1^+ \mathbf{q}_2 \quad (2)$$

或者

$$\mathbf{q}_1 \cdot \mathbf{q}_2 = \mathbf{q}_2^\oplus \mathbf{q}_1. \quad (3)$$

① 假设三维空间内存在坐标系 A. 经过旋转后，变为坐标系 B. 这里 A, B 均为三维。
那么 A, B 两坐标系的单位正交向量为：

$$\vec{e}_A = [e_{Ax}, e_{Ay}, e_{Az}]$$

$$\vec{e}_B = [e_{Bx}, e_{By}, e_{Bz}]$$

另外，假设在空间内存在点 P. 在旋转后 P 点的位置不变，但相对子前后的状态坐标系的坐标分别为 P_A, P_B . 此时满足

$$\vec{e}_A P_A = \vec{e}_B P_B$$

$$\text{那么 } \vec{e}_A^T \vec{e}_A P_A = \vec{e}_B^T \vec{e}_B P_B \Rightarrow P_A = \vec{e}_A^T \vec{e}_B P_B = R_{AB} P_B.$$

这里 R_{AB} 为从坐标系 A 到坐标系 B 而旋转变换关系。

$$R_{BA} = R_{AB}^{-1}. \quad \text{可以由 B 变到 A.}$$

² 若行列式为-1，通常称为瑕旋转 (improper rotation，对应物理当中旋转 + 镜像)。

$$\text{对称式: } \vec{e}_A P_A = \vec{e}_B P_B$$

$$\vec{e}_B^T \vec{e}_A P_A = \vec{e}_B^T \vec{e}_B P_B$$

$$\Rightarrow P_B = \vec{e}_B^T \vec{e}_A P_A = R_{BA} \cdot P_A = R_{AB}^{-1} P_A$$

$$\Rightarrow \begin{cases} P_A = R_{AB} P_B \\ P_B = R_{BA} P_A = R_{AB}^{-1} P_A \end{cases}$$

$$(R_{AB})^T = (\vec{e}_A^T \vec{e}_B)^T = \vec{e}_B^T (\vec{e}_A^T)^T = \vec{e}_B^T \vec{e}_A$$

$$R_{BA} = \vec{e}_B^T \vec{e}_A$$

$$\Rightarrow (R_{AB})^T = R_{BA} = R_{AB}^{-1}$$

$$\Rightarrow \text{对于旋转变换矩阵 } R^T = R^{-1} \Rightarrow RR^T = I.$$

\Rightarrow 逆变换矩阵为正交矩阵.

(2) 对于四元数 q , 虚部为 ε , 实部为 η . $q = (\eta, \varepsilon)$.

ε 为三向量. η 不为 0.

$$(3) \quad q^+ = \begin{bmatrix} \eta & -\varepsilon^x & \varepsilon \\ -\varepsilon^y & \eta & 0 \\ -\varepsilon^z & 0 & \eta \end{bmatrix}$$

$$q^\oplus = \begin{bmatrix} \eta & \varepsilon^x & \varepsilon \\ \varepsilon^y & \eta & 0 \\ \varepsilon^z & 0 & \eta \end{bmatrix}$$

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

5 罗德里格斯公式的证明 (2 分, 约 1 小时)

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 \mathbf{R} 为:

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

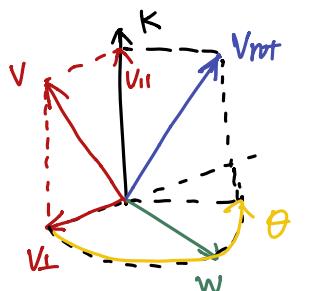
我们在课程中仅指出了该式成立, 但没有给出证明。请你证明此式。

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

罗德里格斯公式存在的基础:

在空间内, 由 A 坐标系旋转到 B 坐标系, 需要一个 \mathbf{R} 的旋转矩阵, 这里 \mathbf{R} 是标准三阶正交矩阵, 所以 \mathbf{R} 的自由度为 3, 说明至少用三个变量来描述 \mathbf{R} .

罗德里格斯公式先确定一个三维单位向量 $\mathbf{k} = [k_x \ k_y \ k_z]^T$ (两个自由度) 和一个标量 θ (一个自由度).



这里 \mathbf{v} 是原向量, \mathbf{k} 是旋转轴, θ 是旋转角度.

$$\mathbf{k} = [k_x \ k_y \ k_z]^T$$

\mathbf{v} 在绕着 \mathbf{k} 旋转 θ 度后, 变成了 \mathbf{v}_{rot} .

$\mathbf{v}_{||}$: \mathbf{v} 与 \mathbf{k} 点积, 即 \mathbf{v} 投影在 \mathbf{k} 方向的向量 $\mathbf{v}_{||}$

$$\mathbf{v}_{||} = (\mathbf{v} \cdot \mathbf{k}) \mathbf{k}$$

\mathbf{v}_{\perp} : \mathbf{v} 投影到平面上且垂直于 $\mathbf{v}_{||}$ 的向量.

$$\mathbf{v}_{\perp} = \mathbf{v} - \mathbf{v}_{||} = \mathbf{v} - (\mathbf{v} \cdot \mathbf{k}) \mathbf{k} = -\mathbf{k} \times (\mathbf{k} \times \mathbf{v})$$

\mathbf{w} : 与 $\mathbf{v}_{||}$, \mathbf{v}_{\perp} 正交构成坐标系的向量

$$\mathbf{w} = \mathbf{k} \times \mathbf{v}.$$

因此我们得到了一组互相正交的坐标系. $\mathbf{v}_{||}$, \mathbf{v}_{\perp} , \mathbf{w} .

$$\Rightarrow \mathbf{v}_{rot} = \mathbf{v}_{||} + \mathbf{v}_{\perp} \cos \theta + \mathbf{w} \sin \theta \quad \rightarrow \mathbf{k} \text{ 为 } \mathbf{K} \text{ 的叉积矩阵.}$$

这时我们引入叉积: $\mathbf{k} = [k_x \ k_y \ k_z]^T$, \mathbf{K} 为一个 skew symmetric matrix.

$$\mathbf{K} = \begin{bmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{bmatrix} \quad \mathbf{k} \times \mathbf{v} = \mathbf{K} \mathbf{v}.$$

之后，我们想用 $v \cdot k$ 来表示 v_{rot} .

$$v_{||} = v - v_{\perp} = v + k \times (k \times v)$$

$$v_{\text{rot}} = v_{||} + v_{\perp} \cos \theta + w \sin \theta$$

$$= v + k \times (k \times v) - k \times (k \times v) \cos \theta + k \times v \sin \theta$$

根据叉积矩阵的性质，我们得出：

$$v_{\text{rot}} = v + (1 - \cos \theta) k^2 v + \sin \theta k v$$

$$v_{\text{rot}} = (I + ((1 - \cos \theta) k^2 + \sin \theta k)) v$$

如果这里 v_{rot} 是旋转后的坐标系 B， v 是旋转前原始的坐标系 A，那么：

$$B = (I + ((1 - \cos \theta) k^2 + \sin \theta k)) A.$$

$$\Rightarrow R = I + ((1 - \cos \theta) k^2 + \sin \theta k)$$

6 四元数运算性质的验证 (1 分, 约 1 小时)

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 \mathbf{q} 旋转点 \mathbf{p} 时，结果为：

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}. \quad (5)$$

我们说，此时 \mathbf{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： $\mathbf{p}' = \mathbf{Q}\mathbf{p}$ 。请根据你的推导，给出矩阵 \mathbf{Q} 。注意此时 \mathbf{p} 和 \mathbf{p}' 都是四元数形式的变量，所以 \mathbf{Q} 为 4×4 的矩阵。

提示：如果使用第 4 题结果，那么有：

$$\begin{aligned} \mathbf{p}' &= \mathbf{q}\mathbf{p}\mathbf{q}^{-1} = \mathbf{q}^+ \mathbf{p}^+ \mathbf{q}^{-1} \\ &= \mathbf{q}^+ \mathbf{q}^{-1\oplus} \mathbf{v}. \end{aligned} \quad (6)$$

从而可以导出四元数至旋转矩阵的转换方式：

$$\mathbf{R} = \mathbf{q}^+ \mathbf{q}^{-1\oplus}. \quad (7)$$

这里，用四元数来旋转点 \mathbf{p} 。我们要将点 \mathbf{p} 表示成四元数而形式。如果点 \mathbf{p} 在坐标系内有坐标为 (x, y, z) ，那么将其表示为四元数即实部为 0， x, y, z 分别为三个虚部的值。

$$\text{所以: } \mathbf{q} = [s, \mathbf{v}_q] \quad \mathbf{p} = [0, \mathbf{v}_p]$$

$$\mathbf{p}' = \mathbf{q} \mathbf{p} \mathbf{q}^{-1}$$

这里 $\mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2} = \frac{[s, -\mathbf{v}_q]^T}{\sqrt{s^2 + v_q^2}} = \left[\frac{s}{\sqrt{s^2 + v_q^2}}, \frac{-\mathbf{v}_q}{\sqrt{s^2 + v_q^2}} \right]^T$

那么

$$\mathbf{q}\mathbf{p} = [s, \mathbf{v}_q] [0, \mathbf{v}_p] = [0 - \mathbf{v}_q^T \mathbf{v}_p, s\mathbf{v}_p + \mathbf{v}_q \times \mathbf{v}_p]^T$$

$$\mathbf{q}\mathbf{p}\mathbf{q}^{-1} = \left[\frac{s}{\sqrt{s^2 + v_q^2}}, \frac{-\mathbf{v}_q}{\sqrt{s^2 + v_q^2}} \right] \left[0 - \mathbf{v}_q^T \mathbf{v}_p, s\mathbf{v}_p + \mathbf{v}_q \times \mathbf{v}_p \right]$$

$$= \left[\frac{s(-\mathbf{v}_q^T \mathbf{v}_p)}{\sqrt{s^2 + v_q^2}} + \left(\frac{\mathbf{v}_q}{\sqrt{s^2 + v_q^2}} \right)^T (s\mathbf{v}_p + \mathbf{v}_q \times \mathbf{v}_p), \dots \right]$$

这里结果因元数而失却为：

$$\begin{aligned}& \frac{s(-v_q^T v_p)}{\sqrt{s^2 + v_q^2}} + \left(\frac{v_q}{\sqrt{s^2 + v_q^2}} \right)^T (s v_p + v_q \times v_p) \\&= \frac{v_q \cdot v_q \times v_p}{\sqrt{s^2 + v_q^2}} \\&= 0.\end{aligned}$$

7 * 熟悉 C++11 (2 分, 约 1 小时)

请注意本题为附加题。

C++ 是一门古老的语言，但它的标准至今仍在不断发展。在 2011 年、2014 年和 2017 年，C++ 的标准又进行了更新，被称为 C++11, C++14, C++17。其中，C++11 标准是最重要的一次更新，让 C++ 发生了重要的改变，也使得近年来的 C++ 程序与你在课本上（比如谭浩强）学到的 C++ 程序有很大的不同。你甚至会惊叹这是一种全新的语言。C++14 和 C++17 则是对 11 标准的完善与扩充。

越来越多的程序开始使用 11 标准，它也会让你在写程序时更加得心应手。本题中，你将学习一些 11 标准下的新语法。请参考本次作业 books/ 目录下的两个 pdf，并回答下面的问题。

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：

```
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}; // 第 15 行
16     std::sort(avec.begin(), avec.end(), [] (const A&a1, const A&a2) {return a1.index<a2.index;}); // 第 16 行
17     for (auto&a: avec) cout<<a.index<<" ";
18     cout<<endl;
19     return 0;
20 }
```

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。

- ① 第 14 行，用了 auto 来自动推导类型，即不用声明遍历的 a 的类型，而是由程序自动推导。
- ② 第 15 行：用了序列 for 循环，可以用于遍历数组、容器、string 以及由 begin 和 end 定义的序列。（有 iterator）
- ③ 第 15 行：vector<A> 的初始化方法。
- ④ 第 16 行：引用了 Lambda 表达式。