

一、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?

回答: 系数矩阵 A 为非奇异矩阵, 等价于:

- A 可逆
- A 行列式不为零
- A 满秩

2. 高斯消元法的原理是什么?

回答:

本质是利用线性方程组的初等变换。

大致分以下两步:

1. 加减消元为上三角矩阵

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \dots & u_{1,n} \\ & u_{2,2} & u_{2,3} & \dots & u_{2,n} \\ \vdots & & \ddots & \ddots & \vdots \\ & (0) & & \ddots & u_{n-1,n} \\ 0 & & \dots & & u_{n,n} \end{bmatrix}$$

2. 回代求未知数

对于 n 阶矩阵, 高斯消元法的算法复杂度是 $O(n^3)$

3. QR 分解的原理是什么?

回答:

将矩阵分解为一个正交矩阵 Q 和一个上三角矩阵 R 相乘。常用于求解线性最小二乘问题。

4. Cholesky 分解的原理是什么?

回答:

将正定的 Hermite 矩阵分解为下三角矩阵与其共轭转置矩阵的乘积

Hermite矩阵是实对称矩阵推广到虚数领域，或者说实对称矩阵是Hermite矩阵的一个特例。

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。

回答:

useEigen.cpp程序如下:

```
#include <iostream>
using namespace std;
#include <Eigen/Core>
#include <Eigen/Dense>
#include <Eigen/Cholesky>

#define SIZE 100
int main() {
    Eigen::Matrix<double, SIZE, SIZE> A;
    A = Eigen::MatrixXd::Random(SIZE, SIZE);

    Eigen::Matrix<double, SIZE, 1> b;
    b = Eigen::MatrixXd::Random(SIZE, 1);

    // 为顺利测试cholesky, 需要对矩阵A进行一些处理
    A = A.array().abs();
    A = A.transpose()*A;

    //inverse
    clock_t time_start = clock();
    Eigen::Matrix<double, SIZE, 1> x1;
    x1 = A.inverse() * b;
    cout<<"Direct Inverse Result:\n"<<x1.matrix().transpose()<<endl;

    cout<<"Time cost:\n"<<(clock() - time_start)*1000/(double)CLOCKS_PER_SEC<<endl;

    //QR
    time_start = clock();
    Eigen::Matrix<double, SIZE, 1> x2;
    x2 = A.fullPivHouseholderQr().solve(b);
    cout<<"QR Result:\n"<<x2.matrix().transpose()<<endl;
    cout<<"Time cost:\n"<<(clock() - time_start)*1000/(double)CLOCKS_PER_SEC<<endl;

    //Cholesky
    time_start = clock();
    Eigen::Matrix<double, SIZE, 1> x3;
    x3 = A.ldlt().solve(b);
    cout<<"Cholesky Result:\n"<<x3.matrix().transpose()<<endl;
    cout<<"Time cost:\n"<<(clock() - time_start)*1000/(double)CLOCKS_PER_SEC<<endl;

    return 0;
}
```

程序运行结果:

Direct Inverse Result:

```
-1400.78 2093.9 -1389.89 -1157.59 749.799 -1086.9 2021.59 3920.96 2324.94 -2401.53 -1092.26 -603.497 -1612.35 -1532.32 -922.897
2589.88 -4185.74 682.045 880.193 -297.364 1721.42 -2022.42 95.7943 -146.585 -2876.39 319.867 -697.67 -1829.4 1146.51 -2336.38
-2414.12 103.986 313.256 269.15 185.677 -1360.64 1118.69 599.454 -749.179 339.669 -386.039 -258.131 1385.33 -152.788 350.1
-2040.69 1330.76 861.132 -116.875 -1319.51 253.054 -38.03 -1530.53 1691.4 1638.2 -26.9992 847.769 -392.561 -1771.23 260.495
-27.4616 1321.63 -937.273 53.049 1591.46 372.121 531.114 1638.39 -2770.32 1415.5 -548.51 -466.825 912.772 -1020.41 -118.177
-63.2475 -851.53 1452.34 -2137.87 1219.31 -1045.09 1319 2358.8 -731.543 -2733.19 1468.79 556.635 858.01 312.771 2115.42
1877.37 -299.188 611.705 967.091 144.038 -598.192 -178.873 1634.85 133.87 19.0554
```

Time cost:

1390.95

QR Result:

```
-1400.78 2093.9 -1389.89 -1157.59 749.799 -1086.9 2021.59 3920.96 2324.94 -2401.53 -1092.26 -603.497 -1612.35 -1532.32 -922.897
2589.88 -4185.74 682.045 880.193 -297.364 1721.42 -2022.42 95.7943 -146.585 -2876.39 319.867 -697.67 -1829.4 1146.51 -2336.38
-2414.12 103.986 313.256 269.15 185.677 -1360.64 1118.69 599.454 -749.179 339.669 -386.039 -258.131 1385.33 -152.788 350.1
-2040.69 1330.76 861.132 -116.875 -1319.51 253.054 -38.03 -1530.53 1691.4 1638.2 -26.9992 847.769 -392.561 -1771.23 260.495
-27.4616 1321.63 -937.273 53.049 1591.46 372.121 531.114 1638.39 -2770.32 1415.5 -548.51 -466.825 912.772 -1020.41 -118.177
-63.2475 -851.53 1452.34 -2137.87 1219.31 -1045.09 1319 2358.8 -731.543 -2733.19 1468.79 556.635 858.01 312.771 2115.42
1877.37 -299.188 611.705 967.091 144.038 -598.192 -178.873 1634.85 133.87 19.0554
```

Time cost:

44.5

Cholesky Result:

```
-1400.78 2093.9 -1389.89 -1157.59 749.799 -1086.9 2021.59 3920.96 2324.94 -2401.53 -1092.26 -603.497 -1612.35 -1532.32 -922.897
2589.88 -4185.74 682.045 880.193 -297.364 1721.42 -2022.42 95.7943 -146.585 -2876.39 319.867 -697.67 -1829.4 1146.51 -2336.38
-2414.12 103.986 313.256 269.15 185.677 -1360.64 1118.69 599.454 -749.179 339.669 -386.039 -258.131 1385.33 -152.788 350.1
-2040.69 1330.76 861.132 -116.875 -1319.51 253.054 -38.03 -1530.53 1691.4 1638.2 -26.9992 847.769 -392.561 -1771.23 260.495
-27.4616 1321.63 -937.273 53.049 1591.46 372.121 531.114 1638.39 -2770.32 1415.5 -548.51 -466.825 912.772 -1020.41 -118.177
-63.2475 -851.53 1452.34 -2137.87 1219.31 -1045.09 1319 2358.8 -731.543 -2733.19 1468.79 556.635 858.01 312.771 2115.42
1877.37 -299.188 611.705 967.091 144.038 -598.192 -178.873 1634.85 133.87 19.0554
```

Time cost:

6.97

程序关键处说明:

1. 因为cholesky分解要求矩阵为正定阵,所以我在程序中对随机生成的100维矩阵做了处理,先变成全为正数的矩阵,然后与其转置相乘,这样可以保证矩阵的正定!
2. 三者求出来的解结果完全相同。同样的矩阵处理时间如下: 正常求逆用时: 1390.05ms; QR分解用时: 44.5ms; cholesky用时6.97ms

二、几何运算练习

设有小萝卜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: $q_1 = [0.55, 0.3, 0.2, 0.2]$, $t_1 = [0.7, 1.1, 0.2]^T$ (q 的第一项为实部)。这里的 q 和 t 表达的是 T_{cw} , 也就是世界到相机的变换关系。小萝卜二号的位姿为 $q_2 = [-0.1, 0.3, -0.7, 0.2]^T$, $t_2 = [-0.1, 0.4, 0.8]^T$ 。现在,小萝卜一号看到某个点在自身的坐标系下,坐标为 $p_1 = [0.5, -0.1, 0.2]^T$, 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。

回答:

```
#include <iostream>
#include <Eigen/Core>
#include <Eigen/Geometry>

using namespace std;
using namespace Eigen;

int main() {

    // 定义变换矩阵T1
    Isometry3d T_C1_W = Isometry3d::Identity();
    Quaterniond q_C1_W(0.55,0.3,0.2,0.2);
    T_C1_W.rotate(q_C1_W.normalized().toRotationMatrix());
    T_C1_W.pretranslate(Vector3d(0.7,1.1,0.2));

    cout<<"一号变换矩阵 \n"<<T_C1_W.matrix()<<endl;

    // 定义变换矩阵T2
    Isometry3d T_C2_W = Isometry3d::Identity();
    Quaterniond q_C2_W(-0.1,0.3,-0.7,0.2);
```

```

T_C2_W.rotate(q_C2_W.normalized().toRotationMatrix());
T_C2_W.pretranslate(Vector3d(-0.1,0.4,0.8));

cout<<"二号变换矩阵 \n"<<T_C2_W.matrix()<<endl;

Vector3d p_C1(0.5, -0.1, 0.2);
Vector3d p_C2 = T_C2_W * T_C1_W.inverse() * p_C1;
cout<<"p_C2: "<<p_C2.transpose()<<endl;

return 1;

}

```

程序运行结果:

```

一号变换矩阵
0.661376  -0.21164  0.719577      0.7
0.719577  0.449735 -0.529101      1.1
-0.21164  0.867725 0.449735      0.2
          0        0        0        1
二号变换矩阵
-0.68254 -0.603175 0.412698      -0.1
-0.730159 0.587302 -0.349206      0.4
-0.031746 -0.539683 -0.84127      0.8
          0        0        0        1
p_C2:  1.08228 0.663509 0.686957

```


三、旋转的表达

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

回答:

1. 设两个坐标基分别是 $[e_1, e_2, e_3]^T$, $[e'_1, e'_2, e'_3]^T$,
 我们根据<<十四讲>>知: 旋转矩阵 $R = \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix} \cdot [e'_1, e'_2, e'_3]$

所以 $R^T R = \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix} \cdot [e_1, e_2, e_3] \cdot \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix} \cdot [e'_1, e'_2, e'_3]$



 I

$= \begin{bmatrix} e_1^T \\ e_2^T \\ e_3^T \end{bmatrix} \cdot [e'_1, e'_2, e'_3] = I$

行列式证明:

$$|R^T R| = |R^T| \cdot |R| = |R| \cdot |R| = 1$$

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。

回答:

其中虚部 ϵ 是三维, 实部 η 为1维

3. 定义如下运算:

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

证明对于两个四元数 q_1, q_2 , 四元数的乘法可以写成: $q_1 \cdot q_2 = (q_1^+)^+ \cdot q_2$

回答:

3. 证明 $q_1 \cdot q_2 = q_1^+ q_2$

等式左侧 $q_1 \cdot q_2 = [h_1 \varepsilon_2 + h_2 \cdot \varepsilon_1 + \varepsilon_1 \times \varepsilon_2, h_1 h_2 - \varepsilon_1^T \cdot \varepsilon_2]$

根据定义的运算, 等式右侧

$$q_1^+ q_2 = \begin{bmatrix} h_1 I + \varepsilon_1^\times & \varepsilon_1 \\ -\varepsilon_1^T & h_1 \end{bmatrix} \begin{bmatrix} \varepsilon_2 \\ h_2 \end{bmatrix}$$

$$= \begin{bmatrix} h_1 \varepsilon_2 + \varepsilon_1^\times \varepsilon_2 + \varepsilon_1 \cdot h_2 \\ h_1 h_2 - \varepsilon_1^T \cdot \varepsilon_2 \end{bmatrix}$$

虚部 ③

实部 ④

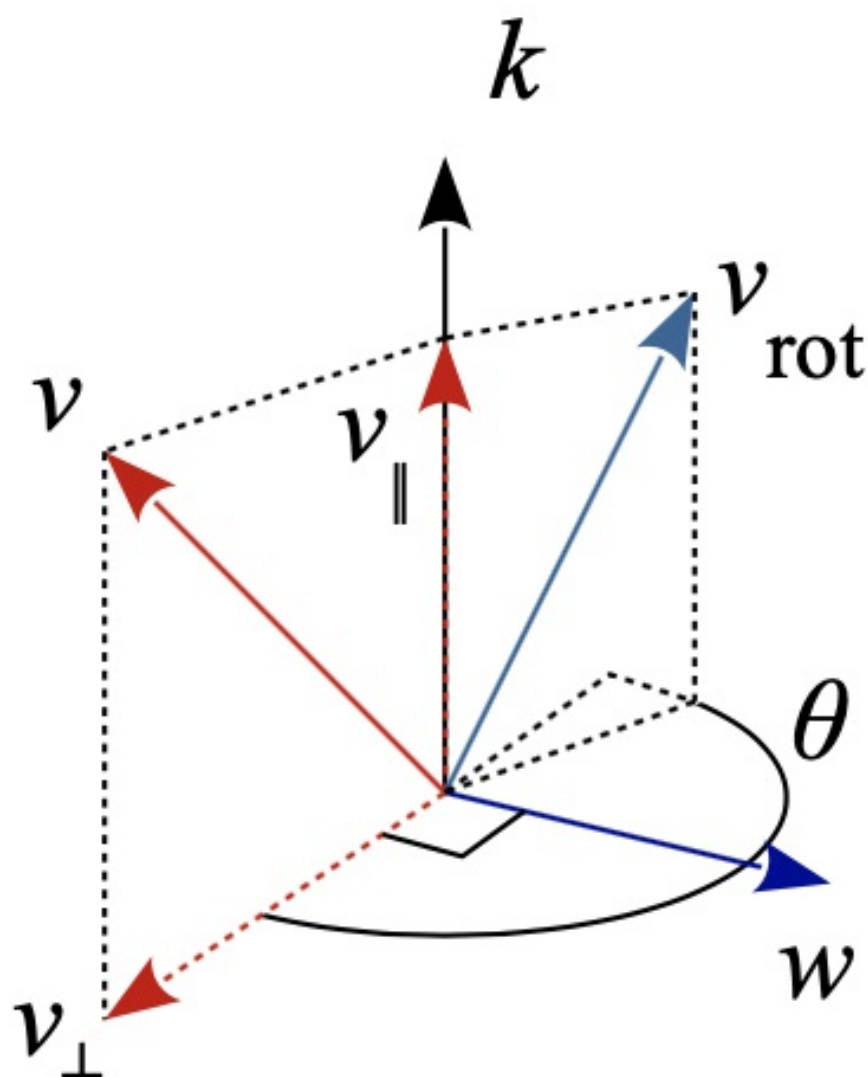
可知: ① = ③ 虚部相同, ② = ④ 实部相同.

故等式成立.

证毕

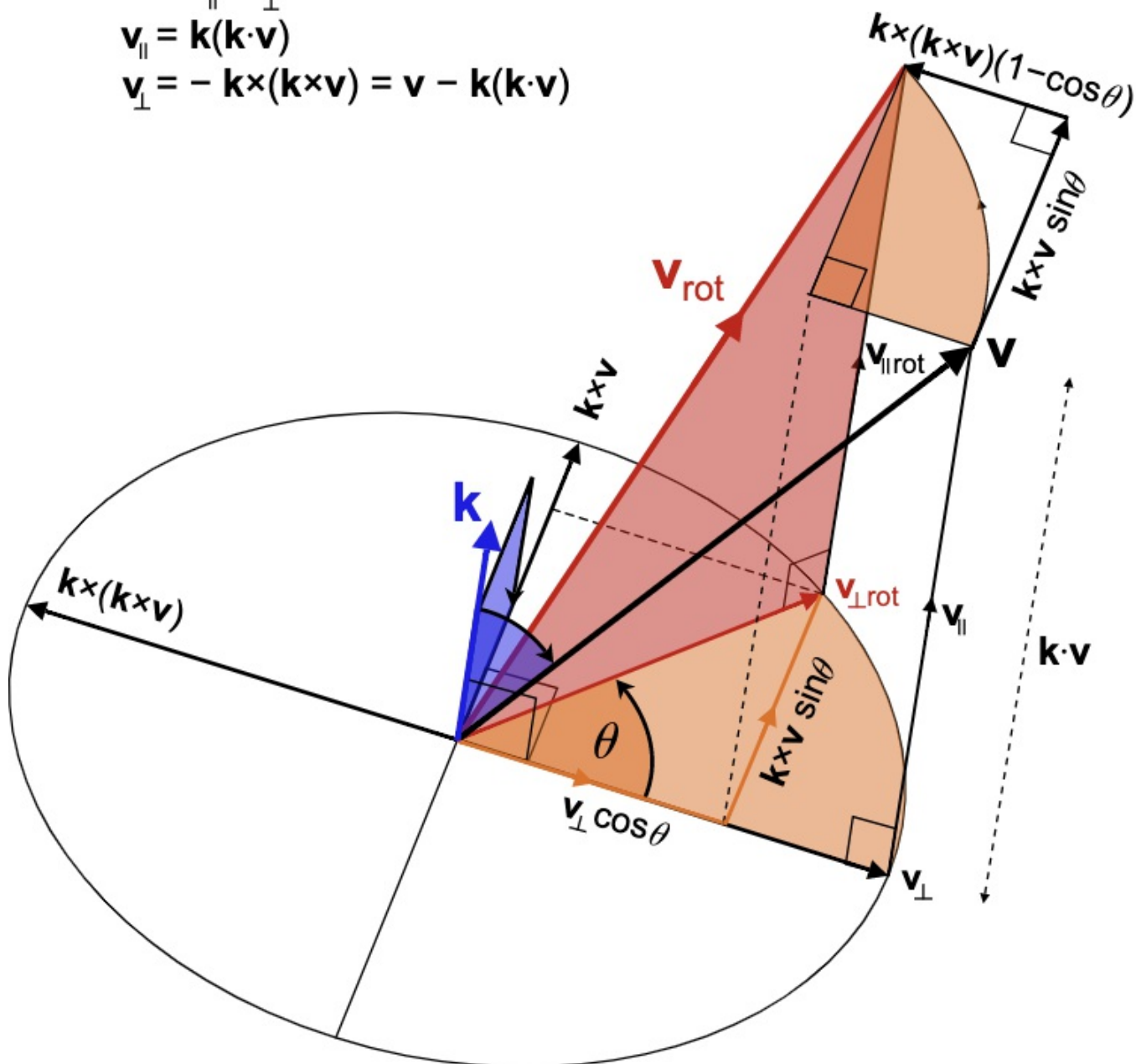
四、罗德里格斯公式证明

回答:



上图表式 \vec{v} 沿 \vec{k} 旋转轴 θ 角到 \vec{v}_{rot}
 \vec{v} 可以分解为 v_{\perp} 和 v_{\parallel} 两部分.
 $\vec{v} = \vec{v}_{\perp} + \vec{v}_{\parallel}$ 其中:
 $\vec{v}_{\perp} = \vec{v} - \vec{v}_{\parallel} = \vec{v} - (\vec{k} \cdot \vec{v}) \cdot \vec{k}$

$$\begin{aligned}\mathbf{v} &= \mathbf{v}_{\parallel} + \mathbf{v}_{\perp} \\ \mathbf{v}_{\parallel} &= \mathbf{k}(\mathbf{k} \cdot \mathbf{v}) \\ \mathbf{v}_{\perp} &= -\mathbf{k} \times (\mathbf{k} \times \mathbf{v}) = \mathbf{v} - \mathbf{k}(\mathbf{k} \cdot \mathbf{v})\end{aligned}$$



同样, 根据上图表示可知:

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

再根据三角函数知识:

$$\vec{v}_{\perp \text{rot}} = \cos \theta \vec{v}_{\perp} + \sin \theta \cdot \vec{k} \times \vec{v}_{\perp}$$

因为 \vec{k} 与 \vec{v}_{\perp} 平行 故上式可以化简成:

由上式可知 $\vec{V}_{\perp \text{rot}}$ 与 $\vec{V}_{\parallel \text{rot}}$ 垂直

$$\vec{V}_{\perp \text{rot}} = \cos\theta \vec{V}_{\perp} + \sin\theta \vec{k} \times \vec{V}$$

所以:

$$\vec{V}_{\text{rot}} = \vec{V}_{\parallel \text{rot}} + \vec{V}_{\perp \text{rot}}$$

$$= V_{\parallel} + \cos\theta \cdot \vec{V}_{\perp} + \sin\theta (\vec{k} \times \vec{V})$$

$$= \cos\theta \vec{V} + (1 - \cos\theta)(\vec{k} \cdot \vec{V})\vec{k} + \sin\theta (\vec{k} \times \vec{V})$$

上式是罗德里格斯矢量表达式, 将 \vec{k} 做反对称矩阵后代入 \vec{k}

可以推导出:

$$R = \cos\theta I + (1 - \cos\theta) \vec{k} \vec{k}^T + \sin\theta \vec{k}^{\wedge}$$

五、四元数运算性质的验证

回答:

① 证明:

$$\text{设 } \mathbf{a} = [s, \vec{v}], \mathbf{b} = [0, \vec{v}']$$

根据四元数乘法:

$$\vec{q}\vec{p} = [-\vec{v}^T \cdot \vec{z}, s \cdot \vec{z} + \vec{v} \times \vec{z}]$$

$$\vec{q}^{-1} = [s, -\vec{v}]$$

故 $q p q^{-1}$ 的实部可以写成:

$$\begin{aligned} & -s \cdot \vec{v}^T \vec{z} + (s \cdot \vec{z} + \vec{v} \wedge \vec{z})^T \vec{v} \\ &= -s \vec{v}^T \vec{z} + s \cdot \vec{z}^T \cdot \vec{v} + \vec{z}^T (\vec{v} \wedge)^T \cdot \vec{v} \end{aligned}$$

$$= 0$$

虚部可以写成:

$$\begin{aligned} & s \cdot (s \cdot \vec{z} + \vec{v} \wedge \vec{z})^\wedge \cdot \vec{z} + \vec{v} \cdot \vec{v}^T \cdot \vec{z} - (s \cdot \vec{z} + \vec{v} \wedge \vec{z})^\wedge \vec{v} \\ &= s^2 \cdot \vec{z} + s \cdot \vec{v} \wedge \vec{z} + \vec{v} \vec{v}^T \vec{z} + \vec{v}^\wedge (s \vec{z} + \vec{v} \wedge \vec{z}) \\ &= (s^2 + 2s \vec{v}^\wedge + |\vec{v}|^2 + \vec{v}^\wedge \vec{v}^\wedge) \vec{z} \end{aligned}$$

综上: R 可以写成:

$$R = \begin{bmatrix} 0 & 0_{1 \times 3} \\ 0_{3 \times 1} & s^2 + 2s \vec{v}^\wedge + |\vec{v}|^2 + \vec{v}^\wedge \vec{v}^\wedge \end{bmatrix}$$

其中右下角是罗德里格斯公式的三维形式。

六、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。

回答:

```
13 | int main() {
14 |     A a1(3), a2(5), a3(9);      vector初始化方法
15 |     vector<A> avec{a1, a2, a3};  lambda表达式
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; 自动转换类型
19 |     return 0;
20 | }
```

for循环迭代