Solution of 第二节课习题 (macOS 平台)

张吉祥

2018 年 3 月 7 日

1 习题说明

2 熟悉 Eigen 矩阵运算

1. 当 A 为非奇异矩阵时，亦即 $\det A \neq 0$ 时，x 有解且唯一。
2. 高斯消元法原理：先把方程组 $Ax = b$ 通过**消元**法变换，等价为上三角方程组，再**回代**求出 x。
3. QR 分解原理：$A = QR$，式中 Q 为正交矩阵，R 为上三角矩阵。故 $Ax = b \Rightarrow QRx = b \Rightarrow Rx = Q^T b$，再**回代**求出 x。
4. Cholesky 分解原理：A 为**对称阵** ($A = A^T$) 时，分解为 $A = LL^T$，式中 L 为下三角矩阵。先求解方程 $Ly = b$ 得出 y，再求解方程 $L^T x = y$ 得出 x，x 即为原方程 $Ax = b$ 的解，此方法又称为平方根法。
5. 程序：

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

#define MATRIX_SIZE 100

int main(int argc, char** argv)
{
    // 构造动态大小的对称矩阵和向量
    Eigen::MatrixXd matrix_A;
    matrix_A = Eigen::MatrixXd::Random(MATRIX_SIZE, MATRIX_SIZE);
    matrix_A = matrix_A.transpose() * matrix_A;
```

```
    Eigen::MatrixXd vector_b;
    vector_b = Eigen::MatrixXd::Random(MATRIX_SIZE, 1);
    // 用 QR 分解求解
    Eigen::MatrixXd QR_x;
    QR_x = matrix_A.colPivHouseholderQr().solve(vector_b);
    // 用 Cholesky 分解求解
    Eigen::MatrixXd Cholesky_x;
    Cholesky_x = matrix_A.ldlt().solve(vector_b);

    cout << "QR_x = " << endl << QR_x << endl;
    cout << "Cholesky_x = " << endl << Cholesky_x << endl;
    return 0;
}
```

注：用 Cholesky 分解求解时，只能针对**对称矩阵**，否则数值错误。

3 几何运算练习

运算结果见图 1：

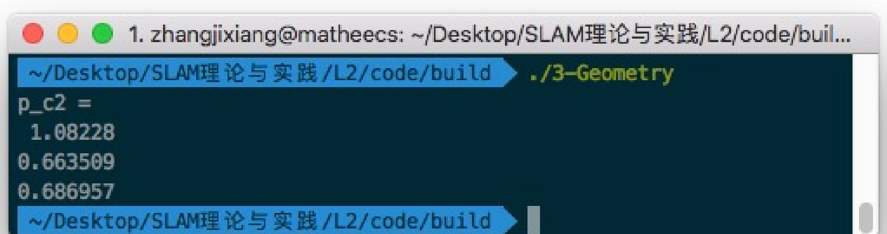

图 1: 几何运算结果

程序：

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

int main(int argc, char** argv)
{
    Eigen::Vector3d p_c1(0.5,-0.1,0.2);
```

```cpp
    Eigen::Vector3d p_c2;

    Eigen::Quaterniond q1(0.55, 0.3, 0.2, 0.2);
    q1.normalize();
    Eigen::Vector3d t1(0.7, 1.1, 0.2);
    Eigen::Isometry3d T_c1w = Eigen::Isometry3d::Identity();
    T_c1w.rotate(q1);
    T_c1w.pretranslate(t1);

    Eigen::Quaterniond q2(-0.1, 0.3, -0.7, 0.2);
    q2.normalize();
    Eigen::Vector3d t2(-0.1, 0.4, 0.8);
    Eigen::Isometry3d T_c2w = Eigen::Isometry3d::Identity();
    T_c2w.rotate(q2);
    T_c2w.pretranslate(t2);

    // 计算该向量在小萝卜二号坐标系下的坐标 p_c2
    // 坐标系转换顺序：小萝卜一号坐标系 -> 世界坐标系 -> 小萝卜二号坐标系
    p_c2 = T_c2w * (T_c1w.inverse()) * p_c1;

    // 输出结果 p_c2
    cout << "p_c2 = \n" << p_c2 << endl;
    return 0;
}
```

4 旋转的表达

1. 证明 (图 2)：记

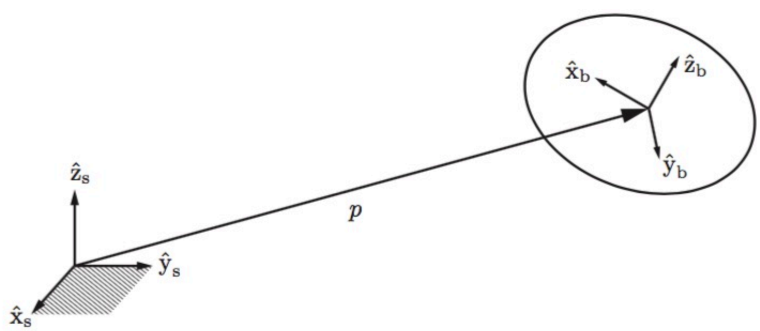

图 2: 旋转矢量推导坐标系

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

根据单位矢量模长条件 (The unit norm condition) 有：

$$\begin{cases} r_{11}^2 + r_{12}^2 + r_{13}^2 = 1 \\ r_{21}^2 + r_{22}^2 + r_{23}^2 = 1 \\ r_{31}^2 + r_{32}^2 + r_{33}^2 = 1 \end{cases} \tag{1}$$

根据单位矢量正交条件 (The orthogonality condition)，即 $\hat{x}_b \cdot \hat{y}_b = \hat{x}_b \cdot \hat{z}_b = \hat{y}_b \cdot \hat{z}_b = 0$，有：

$$\begin{cases} r_{11}r_{12} + r_{21}r_{22} + r_{31}r_{32} = 0 \\ r_{12}r_{13} + r_{22}r_{23} + r_{32}r_{33} = 0 \\ r_{11}r_{13} + r_{21}r_{23} + r_{31}r_{33} = 0 \end{cases} \tag{2}$$

公式 (1) 和 (2) 的矩阵形式等价于

$$R^T R = I \tag{3}$$

故

$$\det R^T R = (\det R)^2 = 1 \Rightarrow \det R = \pm 1 \tag{4}$$

因为我们采用右手系，则取 $\det R = +1$。

2. ε 的维度为 3，η 的维度为 1。

3. 证明：由《视觉 SLAM 十四讲》书中的公式

$$q_a q_b = [s_a s_b - v_a^T v_b, s_a v_b + s_b v_a + v_a \times v_b]$$

故欲证明的等式左边等于

$$q_1 \cdot q_2 = \begin{bmatrix} \eta_1 \varepsilon_2 + \eta_2 \varepsilon_1 + \varepsilon_1 \times \varepsilon_2 \\ \eta_1 \eta_2 - \varepsilon_1^T \varepsilon_2 \end{bmatrix} \tag{5}$$

而等式右边等于

$$q_1^+ q_2 = \begin{bmatrix} \eta_1 \mathbf{1} + \varepsilon_1^\times & \varepsilon_1 \\ -\varepsilon_1^T & \eta_1 \end{bmatrix} \begin{bmatrix} \varepsilon_2 \\ \eta_2 \end{bmatrix} = \begin{bmatrix} \eta_1 \varepsilon_2 + \varepsilon_1^\times \varepsilon_2 + \varepsilon_1 \eta_2 \\ \eta_1 \eta_2 - \varepsilon_1^T \varepsilon_2 \end{bmatrix} \tag{6}$$

或者

$$q_2^\oplus q_1 = \begin{bmatrix} \eta_2 \mathbf{1} - \varepsilon_2^\times & \varepsilon_2 \\ -\varepsilon_2^T & \eta_2 \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \eta_1 \end{bmatrix} = \begin{bmatrix} \eta_1 \varepsilon_2 - \varepsilon_2^\times \varepsilon_1 + \varepsilon_1 \eta_2 \\ \eta_1 \eta_2 - \varepsilon_2^T \varepsilon_1 \end{bmatrix} \tag{7}$$

且 $\varepsilon_1 \times \varepsilon_2 = \varepsilon_1^\times \varepsilon_2 = -\varepsilon_2^\times \varepsilon_1$，所以有

$$q_1 \cdot q_2 = q_1^+ q_2 \tag{8}$$

或者

$$q_1 \cdot q_2 = q_2^\oplus q_1 \tag{9}$$

5 罗德里格斯公式 (Rodrigues' rotation formula) 的证明

$$\mathbf{R} = \cos\theta \mathbf{I} + (1 - \cos\theta)\mathbf{n}\mathbf{n}^T + \sin\theta \mathbf{n}^\wedge \tag{10}$$

式中 $\mathbf{n}$ 为单位矢量。

证明：采用正交分解向量 p(=**a**+**b**) 的方法 (图 3), 分量大小为

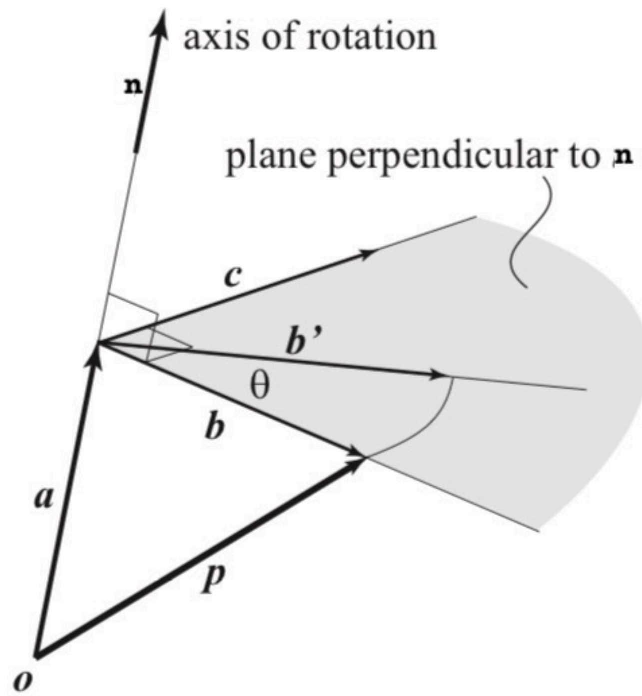

图 3: 罗德里格斯公式推导 $p' = Rp$

$$a = nn^T p \tag{11}$$

$$b = p - a = (1 - nn^T)p \tag{12}$$

从几何角度易知旋转过程中分量 a 不变，而分量 b 旋转到 b'。引入正交于 n, p 的向量 $c = n \times p$，易知 c 的模于 b 相等，则有

$$b' = b\cos\theta + c\sin\theta \tag{13}$$

综上有

$$p' = a + b' = a + b\cos\theta + c\sin\theta = nn^T p + (1 - nn^T)p\cos\theta + n \times p\sin\theta \tag{14}$$

$$= [\cos\theta I + (1 - \cos\theta)nn^T + \sin\theta n^\wedge]p = Rp \tag{15}$$

6 四元数运算性质的验证

虚四元数验证:

$$q=(0.92388,\mathbf{0,\ 0,\ 0.382683}),p=(0,\mathbf{1,\ 0,\ 0})\Rightarrow p'=qpq^{-1}=(0,\mathbf{0.707108,\ 0.707106,\ 0})$$

验证所采用的程序:

```
#include <iostream>
#include <cmath>
using namespace std;

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

int main(int argc, char** argv)
{
    Eigen::Quaterniond q(0.92388, 0, 0, 0.382683);
    q.normalize();

    Eigen::Vector3d v(1, 0, 0);
    Eigen::Quaterniond p;
    p.w() = 0;
    p.vec() = v;
    Eigen::Quaterniond rotatedP = q * p * q.inverse();
    cout<<"(1,0,0) after rotation = \n"<< rotatedP.coeffs() <<endl;
    return 0;
}
```

计算矩阵 Q:

$$p'=qpq^{-1}=qpq^{-1}=q^{+}pq^{-1}=q^{+}p^{+}q^{-1}=q^{+}q^{-1^{\oplus}}p \tag{16}$$

故

$$Q=q^{+}q^{-1^{\oplus}} \tag{17}$$

7 * 熟悉 C++11

说明见注释内容:

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

using namespace std;
```

```cpp

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};
    // lambda 表达式 => [](const A&a1, const A&a2) {return a1.index<a2.index;}
    std::sort(avec.begin(), avec.end(), [](const A&a1, const A&a2) {return a1.index
        <a2.index;});
    // 范围 for 循环 => for ( auto& a: avec )
    // 自动类型推导 => auto&
    for ( auto& a: avec ) cout<<a.index<<" ";
    cout<<endl;
    return 0;
}
```